INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Análise do Controlo de Acesso num Sistema de Gestão de Base de Dados

Luís Miguel Pires Banha Sobral

Beja

2015

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Análise do Controlo de Acesso num Sistema de Gestão de Base de Dados

Dissertação de Mestrado apresentada na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

Elaborado por:

Luís Miguel Pires Banha Sobral

Orientado por:

Prof. Doutora Isabel Sofia Sousa Brito

Beja

2015

Resumo

A sociedade moderna depende cada vez mais da utilização de sistemas informáticos, sendo crucial garantir que os dados que estes sistemas tratam e armazenam estejam protegidos contra fugas de informação. É importante garantir que os dados que são protegidos pelos sistemas somente são utilizados para os fins a que se destinam e por quem de direito. Os sistemas de gestão de bases de dados são os principais guardiões de dados, armazenando e gerindo o acesso a estes dados nos sistemas informáticos. É sobre estes que incidem ameaças e riscos no que respeita a fugas de informação. Nesta tese são apresentados os principais conceitos ao nível do controlo de acesso nos Sistemas de Gestão de Bases de Dados (SGBDs), assim como os modelos de controlo de acesso, o discricionário e o não discricionário e suas características. O Microsoft SQL Server 2012 foi o sistema de gestão de bases de dados escolhido para a implementação dos principais modelos de controlo de acesso tendo como objetivo verificar, através de testes, que características dos modelos de controlo de acesso são suportadas por ele. Com base nesta implementação aplicamos as métricas de avaliação propostas pelo *National Institute of Standards and Technology* (NIST) para análise de segurança nos sistemas de controlo de acesso. Estas métricas, definidas para qualquer sistema de controlo de acesso, foram aplicadas para auditar e servir de guia para a segurança dos processos e operações das empresas. Consideramos assim, que este trabalho pode ser um guia para aqueles que pretendem implementar um sistema de controlo de acesso baseado num dos modelos abordados e que, necessitem de apoio na escolha do sistema de gestão de bases de dados que melhor se adapta às suas necessidades de segurança. Também para aqueles que necessitem de efetuar uma aplicação prática das métricas de avaliação do NIST, dado não existir documentação com exemplos práticos da aplicação das métricas.

Palavras-chave: Sistemas de gestão de bases de dados; Modelos de controlo de acesso; Métricas de avaliação; Segurança da informação; Avaliação de sistemas de controlo de acesso.

Abstract

Modern society relies more and more on information systems, so it is crucial to ensure that the data processed and stored are protected against information leaks. It's important to make sure that the data protected by these systems are only used for its purposes and by its legitimate users. Database management systems are the main repositories of data, storing and managing the access to those data in information systems. The major threats and risks concerning information leaks reside on these systems. In this thesis, it is presented the state of the art of access control in database management systems (DBMS), as well as the main access control models, discretionary and non discretionary and its features. The Microsoft SQL Server 2012 was the database management system chosen to implement the main access control models, so that, through tests, it's verified which access control features are supported by the database management system itself. Based on implemented models, we present and apply evaluation metrics suggested by the National Institute of Standards and Technology, in order to assess the security in access control systems. These metrics, defined for any access control system, were applied to audit and guide the security of enterprises processes and operations. We consider that this paper could be a guide for those who wish to implement an access control system based on one of the addressed models and need some support to choose the database management system best suited to their security requirements. It could also be useful for those who need to perform a pratical execution of NIST evaluation metrics, since there aren't any documents available, using pratical examples.

Keywords: Database management systems; Access control models; Evaluation metrics; Information security; Access Control systems evaluation.

Agradecimentos

Em primeiro lugar queria agradecer à minha orientadora, Professora Doutora Isabel Sofia Sousa Brito, pelo apoio concedido, que sem ele não teria conseguido finalizar esta etapa da minha vida académica.

À minha família, em especial à minha esposa Isabel, pela força dada nos momentos mais difíceis.

A todos aqueles que de uma maneira ou de outra contribuíram para o trabalho aqui apresentado.

Um muito obrigado.

Índice Geral

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas

ACL – *Access Control List*

API - *Application Program Interface*

CRUD - *Create, Retrieve, Update, Delete*

DAC - *Discretionary Access Control*

LBAC - *Lattice Based Access Control*

MAC - *Mandatory Access Control*

MLS - *Multilevel Security*

NIST - *National Institute of Standards and Technology*

RBAC - *Role Based Access Control*

SGBD - Sistema de Gestão de Base de Dados

SoD - *Separation of Duty*

SQL - *Strutured Query Language*

XACML - *eXtensible Access Control Markup Language*

XML – eXtensible Markup Language

1. Introdução

Desde a década de 60 do século passado, que os problemas da segurança informática se começaram a levantar, tendo esta área começado a existir logo na década seguinte. Enormes sistemas de partilha de recursos começaram a existir nos governos, na área militar e em grandes organizações comerciais, onde diversas aplicações, tais como terminais de multibanco, necessitavam de fortes medidas de segurança [1, p. 6].

A segurança da informação toma ainda um papel mais importante quando se pensa que os dados podem ter e ser de vários tipos, inclusive dados considerados sensíveis, quando vistos à luz da lei, nomeadamente da Lei nº 67/98[1]. Atualmente existe uma crescente preocupação com a segurança da informação, não se tendo somente em atenção a confidencialidade, integridade e disponibilidade dessa informação, mas também com o aparecimento de novos requisitos como são a qualidade dos dados, a sua veracidade, pontualidade e proveniência [2, p. 15]. Com o aumento generalizado da informação disponível e, estando esta na maior parte dos casos armazenada em bases de dados, é de enorme importância garantir a segurança no seu acesso.

São estas as bases que sustentam a segurança dos dados e que devem ser mantidas dentro de qualquer organização. Para que tal seja possível é necessário implementar políticas, modelos e mecanismos que se encarreguem de garantir que os dados somente são utilizados para os fins a que se destinam e pelos utilizadores que estão autorizados.

É neste enquadramento que se desenrola o presente estudo, ao dar a conhecer os principais modelos de controlo de acesso e suas características, assim como as suas capacidades de suporte aos requisitos de segurança definidos por métricas de avaliação.

São assim implementados modelos de controlo de acesso com recurso aos mecanismos disponíveis num Sistema de Gestão de Bases de Dados (SGBD) comercial, no caso o MS SQL Server 2012 Developer Edition. A implementação destes modelos de controlo de acesso foi realizado ao longo de vários testes, permitindo analisar como o SGBD escolhido suporta, ou não, os princípios, requisitos e características dos modelos escolhidos. Por outro lado, tornou-se evidente que o controlo de acesso no âmbito dos

[1] Lei da proteção de dados pessoais - http://www.cnpd.pt/bin/legis/nacional/lei_6798.htm

SGBD necessitaria de algo mais para avaliar a segurança do SGBD no suporte destes modelos. Assim, e tendo em vista analisar os conceitos associados ao controlo de acesso suportados pelo MS SQL Server 2012, este é então avaliado de acordo com as métricas propostas pelo (*Guidelines for Access Control System Evaluation Metrics*) *National Institute of Standards and Technology* (NIST) [3]. Concretamente, as funções contempladas nas métricas de avaliação vão então analisar através dos seus itens cada um dos pontos a observar no controlo de acesso gerido pelo SGBD ao nível das bases de dados.

Por fim, são tiradas conclusões das implementações realizadas e das avaliações aos modelos, do ponto de vista da segurança no controlo de acesso e deixados caminhos para continuar com o trabalho aqui apresentado.

1.1. Motivação

Uma base de dados é a melhor forma de armazenar e estruturar informação, sendo hoje utilizadas na maior parte dos sistemas informáticos e estando disponíveis através da *web*. Torna-se aliciante o potencial que existe para exploração nesta área, que é de extrema importância para as organizações e entidades. São cada vez mais usuais os ataques a bases de dados que originam fugas de informações confidenciais, ou noutros casos a sua ocultação, privando até os legítimos donos da informação de acederem a esta. Neste contexto, o controlo de acesso toma ainda maior relevância quando se constata que é na vertente dos privilégios atribuídos aos utilizadores que incidem alguns dos ataques mais críticos efetuados às bases de dados[2].

1.2. Principais Objetivos

Nesta dissertação podemos destacar quatro principais objetivos, onde em primeiro lugar temos o estado da arte ao nível do controlo de acesso nos SGBDs, bem como apresentar dois destes modelos, o discricionário e o não discricionário e suas características.

Como segundo objetivo deste trabalho temos a implementação ao longo de 28 testes destes dois modelos de controlo de acesso, um discricionário e outro não discricionário

[2] http://www.bcs.org/content/ConWebDoc/8852

no MS SQL Server 2012 recorrendo a uma base de dados disponibilizada pela Microsoft para efeitos de testes que é a AdventureWorks2012.

O terceiro principal objetivo consistiu na aplicação prática das métricas propostas pelo NIST aos 28 testes realizados no MS SQL Server 2012 para a implementação dos dois modelos de controlo de acesso, sem recorrer a um ambiente empresarial.

Dos objetivos atrás enunciados, resulta o quarto e último objetivo que é de apoiar os "decisores" na escolha de SGBDs no suporte dos requisitos de segurança que estes pretendem implementar de acordo com as políticas definidas. É então disponibilizado um guia onde se exemplifica a implementação de dois modelos de controlo de acesso no MS SQL Server 2012, uma vez que um terceiro modelo de controlo de acesso não é suportado nativamente por este SGBD.

Resultando num documento que pretende servir de guia para a segurança dos processos e operações das empresas e organizações, devendo estas ainda ser aplicadas em cenários práticos, pré operação, antes de serem implementadas e colocadas em operação.

1.3. Estrutura da Dissertação

A presente dissertação foi estruturada de forma a efetuar uma abordagem inicial sobre o tema, seguindo depois para o estado da arte nesta área, testes realizados e a análise destes, finalizando-se com uma conclusão e propostas de trabalho futuro. De seguida é apresentada a estrutura da dissertação em maior detalhe, ao ser feita uma breve apresentação de cada um dos capítulos e que são:

Introdução – É feita uma introdução ao tema e dado a conhecer o presente trabalho, assim como a motivação e seus principais objetivos.

Estado da Arte – Neste capítulo é realizado um estado da arte num recapitular do enquadramento da importância dos mecanismos de controlo de acesso no âmbito da segurança da informação e alguns conceitos básicos. Também neste capítulo são dados a conhecer três modelos de controlo de acesso e suas principais características, bem como trabalhos relacionados com o tema e seus principais conceitos.

O SGBD MS SQL Server 2012 – É apresentado este sistema de gestão de bases de dados, assim como as edições que estão disponíveis. São dadas a conhecer as principais ferramentas à disposição do administrador para o controlo de acesso, bem como o funcionamento destas.

Controlo de Acesso Discricionário – Neste capítulo mostra-se como é implementado este modelo de controlo de acesso através de uma série de testes no MS SQL Server 2012.

Controlo de Acesso Não Discricionário – Neste capítulo são implementados os modelos de controlo de acesso não discricionário, ou seja, o mandatório e o *Role-Based* através de uma série de testes no MS SQL Server 2012.

Métricas de Avaliação – Neste capítulo são apresentadas as métricas propostas pelo NIST e aplicadas aos testes realizados nos capítulos anteriores de forma a avaliar este SGBD no suporte destas métricas.

Conclusões e Trabalho Futuro – Aqui são apresentadas as contribuições resultantes do presente trabalho e alguns aspetos a ter em consideração, bem como possíveis caminhos para quem queira dar seguimento ao trabalho aqui desenvolvido.

2. Estado da Arte

A evolução dos sistemas informáticos permitiu que estes se integrassem na nossa vida como se dela fizessem parte, armazenando e tratando os nossos dados pessoais, bem como os dados de empresas, organizações e Estados. Não há ainda muito tempo em que os sistemas informáticos eram sistemas isolados e que só estavam acessíveis através dos seus *views*[3] e a quem tinha acesso físico ao sistema, estando muitos destes sistemas ligados entre si com ligações dedicadas e onde mais ninguém tinha acesso.

Com o crescimento da utilização da internet e de todos os serviços que esta faculta, todos os sistemas informáticos estão hoje ligados entre si através desta rede global. Deste facto resultou que os sistemas informáticos e os dados que têm e tratam encontram-se expostos ao mundo, dependendo a sua segurança dos seus próprios sistemas de controlo de acesso. No caso das bases de dados, a sua segurança pode assentar no seu sistema de gestão, sendo este o responsável pela confidencialidade, integridade e disponibilidade dos dados.

Estes sistemas de gestão de bases de dados, responsáveis pela segurança da informação contida nas bases de dados, são sistemas compostos por vários níveis de controlo. São estes sistemas que implementam os modelos de controlo de acesso que por sua vez são definidos com base nas políticas. Assim, neste trabalho são dados a conhecer alguns conceitos que distinguem políticas, modelos e mecanismos de controlo de acesso, e seguidamente são analisados com base em métricas de avalição os mecanismos de controlo de acesso existentes no SGBD MS SQL Server 2012.

[3] Tabelas Virtuais - http://www.w3schools.com/sql/sql_view.asp

2.1. Conceitos de Segurança em Bases de Dados

Antes de mais, é necessário deixar alguns conceitos importantes sobre a segurança dos SGBDs, em particular sobre o controlo de acesso. Assim ficam aqui as definições dos mais importantes e que são:

Modo de Acesso

É uma das características de uma transação num SGBD, podendo o modo de acesso ser de dois tipos, *Read Only* ou *Read Write*. O modo de leitura exclusivo (*Read Only*) indica que o *query* a efetuar é do tipo *select*, dando origem somente a consulta de dados. No caso de o modo de acesso ser do tipo leitura / escrita (*Read / Write*), tanto pode haver consulta como alteração, ou inserção de novos dados [4, p. 770]. Também segundo [2, p. 5], os possíveis modos de acesso que um utilizador pode ter sobre uma tabela são *select* para leitura de dados, *insert* para introdução de dados, *delete* para remover dados na tabela e *update* para alteração de dados.

Controlo de Acesso

É o processo de mediação, entre cada pedido de utilização dos recursos do sistema e dos dados por este mantido, e a determinação de que o pedido realizado, deve ser autorizado ou negado [5, p. 1].

Confidencialidade

Assegura proteger os dados de acessos não autorizados e quando estes são acedidos, que o são por utilizadores autorizados e utilizados somente para fins autorizados. Desta forma protege-se a divulgação da informação que pode violar a privacidade de pessoas ou organizações [6, p. 629].

Integridade

Garante que os dados armazenados se encontram num estado consistente e livre de erros ou anomalias. Mas no caso dos SGBDs, estes têm de garantir não só a integridade dos dados, mas também a dos processos organizacionais, dos utilizadores e dos padrões de utilização [6, p. 629].

Disponibilidade

Refere-se à acessibilidade dos dados aos utilizadores que requerem o seu acesso através de autorizações válidas e para fins autorizados. Esta disponibilidade tem de ser garantida por todo o sistema, para ficar protegido de degradação do serviço ou mesmo da sua interrupção, podendo ser originado por qualquer fonte, podendo esta estar no interior ou no exterior do sistema [6, p. 629].

Os requisitos de segurança atrás mencionados e que no contexto se aplicam a sistemas de gestão de bases de dados, também são os pilares onde assenta a segurança de qualquer outro sistema, que pode ser um sistema operativo ou um sistema de ficheiros. Assim, a segurança da informação armazenada num qualquer sistema, é assegurada por diferentes componentes ou níveis de abstração, que para o nosso caso de estudo, são as políticas, os modelos e os mecanismos de controlo de acesso, a começar pelo nível mais alto de abstração e acabando no mais baixo, onde os mecanismos de controlo de acesso, implementam as respetivas políticas que os definem.

Em seguida, apresentam-se os conceitos relacionados com o controlo de acesso, sendo que estes conceitos representam a base de funcionamento de um sistema de controlo de acesso [3, p. 3].

Objeto: Entidade que armazena e recebe informação, o que pressupõe que o acesso a um determinado objeto também é o acesso à informação que ele contém.

***Subject*:** Uma entidade ativa, que pode ser uma pessoa, um processo ou um dispositivo que cria um fluxo de dados entre objetos.

Operação: Um processo ativo invocado por uma entidade ativa/*Subject*.

Permissão ou Privilégio: É uma autorização para efetuar um tipo de ação no sistema, podendo ser considerada uma combinação entre um objeto e uma operação. Uma única operação aplicada em dois diferentes objetos, representa duas permissões distintas. Da mesma forma, duas operações distintas aplicadas no mesmo objeto, representam duas permissões distintas [1, p. 5]. Neste trabalho iremos usar os dois termos, considerando que ambos têm o mesmo significado.

Lista de Controlo de Acesso: Do inglês *Access Control List* (ACL) é uma lista associada a um determinado objeto e onde constam todas as entidades que têm acesso a esse mesmo objeto, assim como as respetivas permissões. Cada entrada nesta lista é composta por um par (entidade; conjunto de permissões). [3, p. 4]

Capacidades: Do inglês *Capabilities*, que é outro dos controlos de acesso, onde existe uma chave para um objeto específico que em conjunto com o modo de acesso (leitura, escrita, execução) determina que ações pode uma entidade exercer sobre esse objeto. Destas capacidades resulta uma lista de capacidades que pode ser vista como o inverso de uma lista de controlo de acesso, dado que uma ACL é uma lista ligada a um objeto e que define quais as entidades que lhe podem aceder, enquanto a lista de capacidades está ligada a uma entidade e define quais os objetos a que esta pode aceder, formando um par (objeto; modo de acesso). [3, p. 15]

Matriz de Controlo de Acesso: É uma tabela em que cada linha representa uma entidade e cada coluna representa um objeto, onde as entradas são um conjunto de permissões que uma entidade tem sobre um objeto. Uma matriz de controlo de acesso pode ser representada como uma lista de valores triplos constituídos por < entidade, permissões, objetos >. Neste tipo de implementação, a pesquisa de dados onde existe um grande número de valores triplos, torna-se ineficiente e raramente é utilizada. Em vez disso, a matriz de controlo de acesso é subdividida em colunas, dando origem a uma lista de controlo de acesso, descrita acima. Ou então, subdividida em linhas, obtendo-se assim as capacidades, que é uma lista associada a cada utilizador, chamada de lista de capacidades, que indica para cada um dos objetos, os privilégios que a respetiva entidade possui. [3, p. 4]

2.2. Políticas, Modelos e Mecanismos de Controlo de Acesso

As políticas de controlo de acesso são requisitos de alto nível que especificam como é feita a gestão de acessos, e quem, e em que circunstâncias pode aceder a que informação. Estas políticas são implementadas através de mecanismos que tratam do acesso requisitado pelo utilizador e a estrutura existente no sistema. Entre estes dois níveis existe um terceiro que preenche a lacuna existente entre estes dois níveis. É neste nível que se encontra o modelo de controlo de acesso, que faz a ponte entre a política e

o mecanismo de controlo de acesso [7, p. 1]. Os modelos de controlo de acesso também, e segundo [8], descrevem e em alguns casos, provam as características de segurança existente nos sistemas de controlo de acesso.

Em seguida vão ser descritas as principais características das políticas mais conhecidas e "estáveis".

2.2.1. Controlo de Acesso Discricionário

Do inglês *Discretionary Access Control* (DAC), baseada na identidade do utilizador que requisita o acesso e uma série de regras, denominadas de autorizações, que explicitamente definem que ações pode este utilizador realizar sobre os recursos diponíveis [9, p. 3].

Esta política coloca do lado dos utilizadores maior controlo sobre os acessos aos recursos do sistema, onde os objetos são de quem os cria e assim, cada utilizador pode aceder e dar permissões a outros utilizadores sobre os objetos que tem.

Desta forma, um utilizador ao criar uma tabela no sistema, fica automaticamente com todas as permissões sobre esta, e em seguida, pode conceder estas permissões a outros utilizadores. Todas estas ações decorrem sem a supervisão de uma entidade superior.

Esta política passa pela atribuição e revogação de permissões a utilizadores sobre objetos, permissões essas que são dadas pelo dono dos objetos, podendo este também delegar a administração destes objetos [2].

Dentro desta política existem alguns modelos de controlo de acesso, sendo estes:

Modelo de Autorização Sistema R

Este modelo de autorização para bases de dados relacionais contempla dois objetos a proteger, as tabelas e as *views* [2, p. 5]. Os possíveis modos de acesso que um utilizador pode exercer nas tabelas, são as operações SQL permitidas para essas tabelas. Estes modos de acesso facultam permissões para executar comandos para consultas (SELECT), introdução (INSERT), eliminação (DELETE) e modificação (UPDATE) de dados nas tabelas, sendo que para os *views*, estes modos de acesso também se aplicam nos casos onde as suas definições o permitem. A administração de autorizações, neste

modelo, é baseada no conceito de propriedade e na delegação de administração. Dentro deste modelo, cada utilizador devidamente autorizado pode criar uma tabela, ficando dono desta tabela e podendo realizar todos os modos de acesso. O dono da tabela pode ainda delegar permissões a outros utilizadores sobre a tabela, através da concessão de autorizações a esses utilizadores. O utilizador (dono da tabela), pode por sua vez, atribuir permissões a outros utilizadores sobre essa tabela, para que os utilizadores que recebam essas permissões fiquem eles também autorizados a atribuir permissões a outros utilizadores. Esta atribuição de permissões em cadeia cria algumas situações que devem ser tidas em conta, relativamente a possíveis revogações de permissões atribuídas. Temos o caso de um utilizador a quem foi atribuída uma autorização e que por conseguinte atribuiu essa autorização a outros utilizadores, e que posteriormente vê a sua autorização revogada pelo dono da tabela. Nesta situação, todas as autorizações atribuídas pelo utilizador a quem foi revogada a autorização, são também revogadas automaticamente de forma recursiva. Esta revogação de autorizações tem em conta o curso temporal destas, ao analisar as *timestamps*[4] associadas às atribuições de autorizações. Outra característica deste modelo é a implementação da política *Closed World* [2, p. 6], que define que a ausência de autorização deve ser tratada como uma autorização negada. Mas esta ausência de autorização não é impedimento para a atribuição de uma autorização.

Atualmente este modelo de autorização ainda está presente nos SGBDs, com extensões ao modelo original. A inclusão destas extensões deve-se à necessidade de adaptação deste modelo aos requisitos das novas aplicações, onde existem entre outros, novos tipos de objetos a proteger e uma gama mais abrangente de modos de proteção disponíveis nos SGBDs atuais.

Uma das extensões está relacionada com a revogação recursiva de autorizações a utilizadores que as obtiveram de um utilizador que viu a sua autorização revogada, o que em determinadas situações se pode tornar um problema, porque um utilizador pode desejar retirar privilégios a um outro utilizador, mas não a quem esse outro utilizador deu permissões. Para este tipo de revogação de autorizações, há uma operação denominada de *noncascading revoke*, que permite retirar autorizações a um utilizador, mas não a quem este atribuíu. A política *closed world* deste modelo também deu origem

[4] Sequência de caracteres que identifica quando um evento ocorreu.
http://en.wikipedia.org/wiki/Timestamp

a uma das suas extensões, que lida com autorizações negativas, porque o facto de um utilizador não ter autorização para uma determinada operação sobre um objeto (ausência de autorização) não o impede de ver essa autorização concedida. Para que tal impedimento aconteça, foi introduzido o conceito de autorização negativa (negação de autorização), com precedência desta sobre a autorização positiva. Finalmente outra das extensões que se pode destacar é a atribuição de autorizações temporais, que só têm validade por um período de tempo definido. Com uma autorização temporal é possível atribuir permissões por períodos definidos de tempo, sem que o utilizador que concede a autorização tenha de se preocupar em revogá-la posteriormente [2, p. 5].

Modelo de Controlo de Acesso Refinado e Baseado em Conteúdos

Este modelo de controlo de acesso, também utilizado em políticas DAC, baseia a atribuição de autorizações no conteúdo dos dados armazenados [2, p. 7]. É possível, com este modelo, definir regras para os utilizadores em que estes só têm permissões para aceder a tuplos que satisfaçam um determinado requisito definido por uma cláusula *where*. A linguagem SQL, que permite declarar condições para verificação dos conteúdos dos dados, dá suporte ao mecanismo mais utilizado nas bases de dados relacionais para controlo de acesso baseado em conteúdos, os *views*. Dentro destes *views*, existem duas categorias, que são *protection views* e *shortand views*[5], em que os primeiros estão adaptados a dar suporte ao controlo de acesso baseado em conteúdos e os segundos estão direcionados para simplificar a escrita de *queries*. Atualmente existem mecanismos de controlo de acesso que dão suporte ao nível dos tuplos, com *views* personalizados para utilizadores e para tuplos, e que desta forma suportam o controlo de acesso refinado, baseado nos conteúdos dos dados, onde os *views* têm um papel de filtro aos dados devolvidos, aquando da execução de um *query*.

2.2.2. Controlo de Acesso Mandatório

Esta política de controlo de acesso não discricionária, mais conhecida por MAC (do inglês *Mandatory Access Control*), assenta num sistema onde existe uma entidade superior (administrador) que controla o acesso a todos os objetos existentes no sistema de forma geral e onde os utilizadores não podem alterar as regras definidas de forma

[5] E. Bertino and L.M. Haas, "Views and Security in Distributed Database Management Systems," Proc. Int'l Conf. Extending Database Technology, Mar. 1988.

individual [10, p. 695]. Segundo esta política, todos os objetos e utilizadores são colocados em níveis de acesso, onde um utilizador para aceder a um determinado objeto, necessita de ter um nível igual ou superior ao do objeto que pretende aceder. Nesta política os utilizadores são entidades ativas que pretendem aceder aos objetos, que são as entidades passivas que armazenam informação e existindo uma relação entre os utilizadores e os objetos. Esta relação entre o utilizador e o objeto, é definida por um conjunto de classes de acesso, onde uma classe de acesso é constituída por um nível de segurança e por um conjunto de categorias.

Esta política de controlo de acesso tem como modelo associado, o modelo Bell-LaPadula, que pode ser descrito por objetos (informação contida na base de dados), entidades (utilizadores, programas, etc.) e níveis de segurança [10, p. 706]. Esta política é a mais utilizada em bases de dados de alta segurança, como é o caso de sistemas que contenham informações militares e governamentais. Os dados são classificados em níveis de segurança conforme são mais ou menos sensíveis, estando os mais sensíveis no nível *Top Secret* e os menos sensíveis no nível *Unclassified*. Em seguida, são mostradas as relações existentes entre os diferentes níveis de segurança [2, p. 9].

TS (Top Secret) > S (Secret) > C (Confidential) > U (Unclassified)

Sendo que estes níveis de segurança tanto se aplicam aos objetos como aos utilizadores, e é com base nestes níveis que é feito o controlo de acesso dos utilizadores aos objetos. Para que um utilizador possa aceder a um determinado objeto, e dependendo do modo de acesso, tem de o utilizador pertencer a uma classe cuja relação com a classe do objeto seja a seguinte:

Modo de Leitura

classe (Utilizador) ≥ classe (Objeto)

Modo de Escrita

classe (Utilizador) ≤ classe (Objeto)

A primeira regra, utilizada na leitura, é bastante intuitiva, porque um utilizador não pode aceder a informações que pertencem a um nível superior ao seu. Já a segunda regra, de escrita, não é tão intuitiva, uma vez que proíbe que um utilizador de um nível superior

escreva num objeto de nível inferior. Mas ao analisarmos mais detalhadamente, esta regra evita que a informação existente no sistema possa descer nos níveis de segurança, como se pode demonstrar no seguinte exemplo. Um utilizador que pertença ao nível TS pode ler informação neste nível, mas não pode escrever no nível U. Se assim não fosse este utilizador podia copiar um objeto do nível TS para o nível U, permitindo que este objeto ficasse a pertencer ao nível menos sensível de segurança e visível a todo o sistema.

Apesar de a maior parte dos SGBDs comerciais só incorporarem mecanismos de controlo de acesso DAC, estes podem ser implementados em conjunto, sempre que sejam necessários vários níveis de segurança.

2.2.3. *Role Based Access Control*

Outra política de controlo de acesso não discricionária, a *Role Based Access Control* (RBAC) é considerada a norma para a gestão de privilégios em sistemas comerciais e aplicações [11]. Surgiu como alternativa às políticas MAC e DAC, de forma a contornar as limitações destas e a simplificar a administração de privilégios no sistema. Esta política faz a gestão de privilégios com base em *roles*, que são depois atribuídas a utilizadores, funcionando as *roles* como regras de atribuição de privilégios [1, p. 10]. Estas *roles* têm associadas permissões para executar determinadas ações, que ao serem atribuídas aos utilizadores lhes facultam as permissões que lhes estão atribuídas, de forma semelhante às permissões atribuídas a grupos de utilizadores em sistemas operativos.

De acordo com [11] e como se pode ver na Figura 1, o modelo RBAC é constituído por três elementos chave e que são os utilizadores, as *roles* e as permissões.

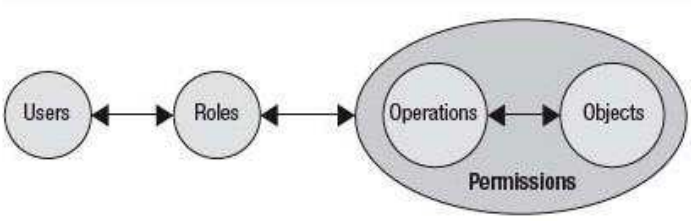

Figura 1 - Modelo RBAC e seus elementos chave [11]

Sendo que:

Utilizadores (*Users*) - São indivíduos que têm um cargo dentro de uma organização e que normalmente desenvolvem funções individuais dentro dessa organização;

Roles - No contexto de negócio, são cargos (ou funções) desempenhados e responsabilidades associadas. As responsabilidades representam os utilizadores que de uma forma explícita ou implícita têm autorização para desempenhar as suas funções laborais. No contexto tecnológico as *roles* representam conjuntos de direitos/autorizações que a pessoa herda (adquire) da perspetiva da aplicação para desempenhar as suas funções;

Permissões - No contexto tecnológico, são a posse de autorizações para que alguém possa efetuar uma operação num objeto controlado pelo modelo RBAC dentro de uma aplicação ou sistema.

A administração das permissões fica em muito simplificada com a possibilidade de se agruparem estas permissões em *roles*, que consoante o grupo a que pertencem, assim permitem atribuir um conjunto de permissões a um utilizador ou vários desse grupo. A administração tem de gerir dois tipos de associações, a associação dos utilizadores às *roles* e a associação das *roles* às permissões.

Este tipo de política de controlo de acesso desde o seu início, assenta em três regras básicas que devem ser respeitadas [1, p. 11], e que são:

Regra 1 - Atribuição da *role*

Um utilizador só pode efetuar uma transação se tiver uma *role* selecionada ou atribuída, sendo que unicamente o processo de identificação e autenticação no sistema não é considerado uma transação, ao contrário de todas as outras atividades efetuadas pelos utilizadores no sistema. Todos os utilizadores ativos no sistema têm de ter pelo menos uma *role* ativa associada.

Regra 2 - Autorização da *role*

Para que uma *role* esteja ativa para um utilizador, tem de estar autorizada para esse utilizador. Em conjunto com a regra anterior, fica garantido que só é atribuído a um utilizador uma *role* para a qual este tem autorização.

Regra 3 - Autorização da transação

Um utilizador só pode realizar uma transação se esta estiver autorizada para a *role* que o utilizador tem ativa. Desta forma, em conjunto com as duas anteriores regras, é assegurado que os utilizadores só podem realizar transações para as quais estão autorizados.

Este tipo de controlo de acesso é bastante flexível em situações de alteração do nível hierárquico de um utilizador dentro de uma organização, facilitando a alteração dos privilégios que o utilizador detém. Com outras políticas como são as MAC e DAC, é necessário atribuir e revogar permissões, uma a uma, ao utilizador, tornando esta tarefa complicada e morosa.

2.3. Mecanismos de Controlo de Acesso nos SGBDs

Os mecanismos de controlo de acesso encontram-se no nível de abstração mais baixo num sistema de controlo de acesso, e são eles que põem em prática as políticas e os modelos definidos.

Estes mecanismos de controlo de acesso garantem a confidencialidade dos dados armazenados no sistema e controlam os acessos dos utilizadores aos recursos disponíveis. Verificam, por exemplo, que um utilizador ao tentar aceder aos dados armazenados, tem de facto permissão para executar essa operação. A atribuição de permissões é dada pela política de controlo de acesso que se encontra implementada no sistema. A integridade dos dados fica também, em parte, assegurada pelos mecanismos de controlo de acesso, ao verificarem se um determinado utilizador tem permissão para modificar os dados que este pretende modificar e sendo, em seguida, realizada uma verificação semântica dos dados alterados. A disponibilidade dos dados encontra-se a

cargo de subsistemas de recuperação de falhas e de gestão de recursos que funcionam em paralelo com o mecanismo de controlo de acesso [2, p. 3].

Estas funções são implementadas pelos SGBDs, sendo estes os responsáveis pela segurança dos dados armazenados no sistema, e que fazem a mediação entre todo e qualquer tipo de acesso aos recursos mantidos pelo sistema, verificando se o pedido de acesso deve ser aprovado ou negado.

Um SGBD gerido por uma autoridade central, que é o seu administrador, tem a responsabilidade de atribuir e revogar privilégios a contas, utilizadores e grupos de utilizadores, nomeadamente, segundo [4, p. 839], através das seguintes ações:

Ação 1. Criação de conta - Esta ação cria uma nova conta e uma nova palavra passe para um utilizador ou grupo de utilizadores, para facultar acesso (autenticação) ao sistema.

Ação 2. Atribuição de privilégios - Esta ação permite ao administrador atribuir certos privilégios a algumas contas.

Ação 3. Revogação de privilégios - Esta ação permite ao administrador cancelar privilégios que foram anteriormente atribuídos a algumas contas.

Ação 4. Atribuição de nível de segurança - Esta ação consiste em atribuir às contas o nível de segurança apropriado.

No caso do ponto número 1, é uma ação que permite controlar os acessos na generalidade, permitindo a autenticação dos utilizadores no sistema. Os pontos 2 e 3 são utilizados em políticas de controlo de acesso discricionário e o ponto 4 destina-se a políticas de controlo de acesso mandatório.

Ao nível das soluções comerciais, existe uma variedade de SGBDs, que passa por sistemas comerciais e sistemas *Open Source*[6].

[6] Baseados em software *open source* - http://en.wikipedia.org/wiki/Open-source_software

2.3.1. Trabalhos Relacionados

Segundo [2] com o aumento da dependência dos sistemas de informação, as organizações começaram a necessitar de implementar medidas de segurança para a gestão dos dados. A exposição das bases de dados à internet, através das aplicações *web*[7] com ligações a estas, tornou a segurança dos dados em algo crucial para a sobrevivência das organizações. Assim, este trabalho mostra os principais conceitos associados à segurança das bases de dados e resume as técnicas conhecidas nesta matéria.

Desde cedo se percebeu que era necessário desenvolver soluções de segurança que vão de encontro a três requisitos básicos, e que são eles:

- Confidencialidade;

- Integridade;

- Disponibilidade.

Estes requisitos são ainda complementados com um outro, que é a privacidade dos dados, e o consentimento dos utilizadores para o uso dos mesmos, de forma a garantir que os dados são usados unicamente para os fins a que se destinam e não outros.

Assim foram desenvolvidos mecanismos que contêm diversos componentes para em conjunto assegurarem a confidencialidade dos dados, para que sempre que um utilizador tente efetuar um acesso aos dados, lhe seja verificada a autorização.

Surgem assim duas classes de modelos de controlo de acesso, uma baseada em políticas discricionárias e outra em políticas mandatórias para os sistemas de gestão de bases de dados relacionais. Estes modelos iniciais, e particularmente o discricionário, introduziram importantes princípios para o controlo de acesso, retirados dos sistemas de ficheiros e dos sistemas operativos. O primeiro destes princípios define que o modelo de controlo de acesso para as bases de dados deveria ser expresso nos termos de um modelo lógico de dados. O segundo princípio define que o modelo deveria suportar controlo de acesso baseado nos conteúdos dos dados.

[7] http://en.wikipedia.org/wiki/Web_application

Mas os modelos de controlo de acesso discricionário tinham uma fraqueza, o não controlar a propagação da informação a partir do momento em que esta é acedida por utilizadores autorizados, deixando estes dados vulneráveis a ataques, tais como atribuição excessiva de permissões ou abuso das permissões[8]. Mas devido à flexibilidade proporcionada por mecanismos que permitem aos utilizadores atribuir permissões a outros utilizadores sobre os seus dados, é que as políticas discricionárias foram adotadas em diversos ambientes aplicacionais, assim como nos SGBDs comerciais. Um dos aspetos relacionados com este tipo de controlo de acesso é a administração de autorizações, onde é possível atribuir e revogar autorizações.

Para os modelos de controlo de acesso discricionário, foi dado um importante contributo através do desenvolvimento do modelo de controlo de acesso *System R*, que veio a influenciar os atuais SGBDs comerciais. Também foram propostas posteriormente algumas características tais como autorização negativa, autorizações baseadas em *roles*, em tarefas e autorizações temporais.

Dentro dos modelos discricionários e um dos primeiros modelos de autorizações a ser desenvolvido para bases de dados relacionais foi o *System R*, onde os objetos a proteger são tabelas e *views*, com os utilizadores a poderem exercer o acesso definido pelo SQL (por exemplo, GRANT). A administração de autorizações neste modelo permite ao dono dos objetos atribuir permissões a outros utilizadores, para que estes possam atribuir e revogar permissões sobre esses objetos, criando situações onde os utilizadores a quem foram dadas permissões, deem eles próprios permissões a outros utilizadores. Assim, ao ser revogada a primeira permissão dada, todas as outras que foram dadas posteriormente serão automaticamente revogadas. Uma extensão a este modelo criou uma opção denominada de *noncascading*, onde as permissões não são revogadas de forma recursiva. Este modelo utiliza a política *closed world*, e dá prioridade à negação das autorizações. Uma outra extensão a este modelo implementa autorizações temporais, onde a autorização só é válida durante um período de tempo pré-definido.

Outro importante requisito dos mecanismos de controlo de acesso é o controlo de acesso *content-based*, que permite filtrar consultas pelos conteúdos dos dados existentes.

[8] http://www.bcs.org/content/ConWebDoc/8852

Os sistemas de controlo mandatório e os de bases de dados multi-nível, do inglês *Multilevel Security* (MLS) tentaram evitar alguns dos problemas dos modelos discricionários ao desenvolver controlos de acesso baseados na classificação da informação, tendo alguns destes sido introduzidos em sistemas de controlo de acesso comerciais. Estes baseiam o controlo de acesso em níveis onde se encontram os objetos e os utilizadores, onde um utilizador não pode escrever em objetos que se encontrem nos níveis abaixo e não pode ler em objetos que se encontrem nos níveis acima. Existem modelos mandatórios de um único nível ou MLS, onde no primeiro caso todos os objetos se encontram no mesmo nível. Estes modelos, inicialmente pensados para fins militares, eram bastante rígidos e vocacionados para ambientes fechados e controlados. Devido à falta de aplicações e ao seu insucesso comercial, que desde há alguns anos atrás o desenvolvimento de SGBDs MLS foi descontinuado por parte das empresas de *software*.

Os modelos RBAC são a mais importante das inovações nos modelos de controlo de acesso, tendo vindo simplificar a administração de autorizações e representando as políticas de controlo de acesso das organizações. As autorizações são atribuidas às *roles* e estas aos utilizadores, não sendo atribuídas as autorizações diretamente aos utilizadores.

Outro ponto relevante no controlo de acesso é a utilização do XML. O XML é o *standard* para documentos e dados que circulam na web, sendo também muito utilizado em aplicações e na indústria. Devido à sua grande utilização, foi necessário desenvolver sistemas de controlo de acesso para o XML, tendo como *standard* o *Extensible Access Control Markup Language* (XACML).

A preservação dos dados é cada vez mais um fator de preocupação e exige técnicas de administração destinadas a manter a privacidade dos dados. Estas técnicas aplicam-se tanto ao nível do armazenamento dos dados como da sua transmissão, uma vez que no primeiro caso é necessário dificultar o uso das sofisticadas técnicas de *data mining*[9] existentes, e no segundo caso é necessário garantir a anonimização dos dados transmitidos.

[9] http://www.statsoft.com/Textbook/Data-Mining-Techniques

De todos estes fatores, acima mencionados, resultam grandes desafios para a proteção dos dados, não só para os tradicionais requisitos de segurança, confidencialidade, integridade e disponibilidade, mas também para novos requisitos, tais como a qualidade e completude dos dados, a atualização destes e a sua proviniência.

Segundo [5], um sistema de controlo de acesso pode ser considerado em três níveis abstratos de controlo e que são a política de controlo de acesso, o modelo de controlo de acesso e o mecanismo de controlo de acesso. Este trabalho mostra as tendências emergentes no campo do controlo de acesso que visam dar suporte aos SGBDs.

Os modelos de controlo de acesso clássicos podem ser agrupados em três classes e que são o discricionário (DAC), mandatório (MAC) e baseado em *roles* (RBAC).

O controlo de acesso discricionário é baseado na identidade do utilizador que solicita o acesso e num conjunto de regras denominadas autorizações que definem quais as ações permitidas nos recursos. O primeiro modelo discricionário proposto foi a matriz de acesso e que pode ser implementada através dos seguintes mecanismos: Tabela de autorizações; Lista de controlo de acesso; Capacidades. Com a evolução dos sistemas de controlo de acesso, estes passaram a ter suporte para algumas características, tais como:

Condições

A validade de uma autorização depende da satisfação de certos requisitos específicos, sendo necessário que os SGBDs suportem condições associadas a autorizações.

Abstrações

De forma a simplificar a definição do processo de autorização, o controlo de acesso discricionário também suporta grupos de utilizadores e classes de objetos que podem ser organizados de forma hierárquica.

Exceções

É necessário que os mecanismos de controlo de acesso suportem o uso de exceções, como em situações onde são atribuídas permissões a um utilizador sobre um

determinado *schema*[10] e em que este não pode ter acesso de escrita a uma certa tabela. Daqui surge a necessidade de utilizar autorizações positivas e negativas.

Para o caso das exceções, foram introduzidas autorizações positivas e negativas (GRANT e DENY), o que deu origem a dois problemas, inconsistências e autorizações incompletas. Para as inconsistências foram adotadas políticas de resolução de conflitos, enquanto para as autorizações incompletas é utilizada uma política por defeito sendo que a mais usual é a *closed world*.

A resolução da ocorrência de exceções, bem como de problemas de inconsistência, depende de qual das políticas de resolução de conflitos é colocada em prática pelo SGBD. Como políticas de resolução de conflitos existem [9, p. 5]:

- Inexistência de conflitos: a ocorrência de um conflito é considerada um erro;

- Negação tem precedência: as autorizações negativas têm prioridade;

- Permissão tem precedência: as autorizações positivas têm prioridade;

- Não existem precedências: os conflitos não são resolvidos;

- O mais específico tem precedência: Uma autorização associada a um elemento ***n*** tem prioridade sobre uma autorização contrária (autorização com a mesma entidade, objeto e ação mas de sinal contrário) associada ao antecessor de ***n*** para todos os sucessores de ***n***.

- O mais específico no caminho tem precedência: uma autorização associada a um elemento ***n*** tem precedência sobre uma autorização contraditória associada ao antecessor de ***n*** para todos os descendentes de ***n***, somente para caminhos que passem por ***n***.

As políticas de segurança mandatórias baseiam o seu controlo de acesso numa autoridade central reguladora, sendo normalmente utilizada uma política MLS onde são colocados os objetos e os utilizadores, criando classes de acesso que dominam umas sobre outras.

A política mandatória baseada no sigilo tem como principal objetivo manter a confidencialidade dos dados, onde o nível de segurança da classe de acesso do objeto

[10] http://www.quackit.com/sql_server/sql_server_2008/tutorial/sql_server_database_schemas.cfm

reflete a sensibilidade dos dados e o nível de segurança da classe de acesso do utilizador, *clearance*, mostra o grau de confiança atribuído ao utilizador. O conjunto de categorias associadas aos utilizadores e aos objetos define a área de competência dos utilizadores sobre os dados.

Sempre que um acesso é requisitado por um utilizador, são aplicados os princípios de não ler no nível superior e não escrever no nível inferior.

A política mandatória baseada na integridade tem como principal objetivo impedir os utilizadores de modificar dados que não possam ser alterados. O nível de integridade associado ao utilizador reflete o grau de confiança depositado no utilizador para escrever e modificar dados sensíveis. O nível de integridade associado ao objeto indica o grau de confiança depositado nos dados armazenados no objeto e os potenciais danos que possam ocorrer de acessos não autorizados.

O modelo de controlo de acesso baseado em *roles* permite ao administrador atribuir permissões às *roles* e depois estas *roles* são atribuídas aos utilizadores. Aos utilizadores pode ser atribuído mais do que uma *role*, e a mesma *role* pode ser atribuída a mais que um utilizador. Neste modelo existem grupos de utilizadores, mas que são diferentes destas *roles*, uma vez que os grupos de utilizadores são conjuntos de utilizadores e as *roles* são conjuntos de permissões. Este modelo está bastante adequado ao meio empresarial, dado que não distingue os utilizadores individualmente, mas pelas suas responsabilidades, refletindo o ambiente das empresas.

Segundo [12], os mecanismos de controlo de acesso, elementos determinantes na correta validação das operações solicitadas aos sistemas de controlo de acesso, devem ser avaliados com base nas métricas definidas pelo NIST. Estas métricas têm como finalidade auditar e servir de guia para a segurança dos processos e operações das empresas, devendo estas ser aplicadas em cenários práticos antes de serem implementadas e colocadas em operação. Deste modo, permitem uma visualização mais abrangente das características de segurança que estão incorporadas no modelo de controlo de acesso, e se este é o mais indicado para a situação que está a ser abordada. Para controlo de acesso em sistemas baseados na nuvem (do inglês *cloud*), são propostas para avaliação dos sistemas de controlo de acesso, as seguintes métricas:

Least Privilege; *Separation of Duty; Management Complexity; Enforcement Mechanism; Policy Conflicts; Horizontal Scope; Configuration Flexibility*.

A computação em nuvem[11] é um exemplo que contém preocupações a nível da segurança, tanto do lado do servidor como do cliente. O servidor ao disponibilizar recursos e serviços, deve-se certificar que estes não são usados por utilizadores ilegais ou maliciosos, bem como os clientes se devem certificar que está assegurada a privacidade dos dados e que o servidor não está comprometido.

O RBAC aplicado à computação em nuvem, é tido como o mais indicado para empresas e organizações onde é fornecido em duas etapas, a primeira delas em que é efetuada a autenticação através dos atributos do utilizador e em seguida a correspondente validação das *roles*. Inicialmente o utilizador é autenticado com as suas credenciais e são identificadas as *roles* que lhe estão atribuídas e as permissões que estão associadas a estas.

Este modelo ao ser analisado com as métricas anteriormente enunciadas, o sistema RBAC suporta totalmente o princípio do *Least Privilege*, uma vez que privilégios são dados de acordo com a especificação das *roles* e os atributos do utilizador.

É parcialmente suportado o *Separation of Duty*, já que os privilégios são dados mais de acordo com as *roles* do que com os atributos/características dos utilizadores.

O *Enforcement Mechanism* envolve dois passos para a validação dos atributos e verificação dos *roles*, estando este sistema devidamente suportado dado a natureza distribuída da nuvem.

Não suporta *Policy Conflits* caso estes ocorram devido a imprecisões na especificação da política. Devido a mediação entre os níveis de APIs, é aumentado o seu *Horizontal Scope* nas diferentes aplicações da *cloud*. A configuração simples das APIs para a vasta gama de aplicações faz com que este sistema suporte a *Configuration Flexibility*.

Neste artigo foram analisados outros sistemas de controlo de acesso para a *cloud*, mas saem do âmbito do presente trabalho, sendo os resultados da aplicação das diferentes métricas analisados numa tabela comparativa com uma escala de avaliação de três

[11] Computação em nuvem – Do inglês *cloud computing*, http://en.wikipedia.org/wiki/Cloud_computing

níveis, que são *High, Medium e Low*. Os níveis utilizados para classificar os resultados obtidos na aplicação das métricas podem ser traduzidos para "Completamente Suportado", "Parcialmente Suportado" e "Não Suportado", ficando desta forma mais intuitivo o seu significado prático.

Os trabalhos apresentados nesta secção serviram de inspiração para esta tese. A nossa proposta difere destes trabalhos no que respeita à verificação e análise dos mecanismos de controlo de acessos num SGBD comercial, fazendo a "ponte" entre os conceitos e a prática.

3. O SGBD MS SQL Server 2012

Neste capítulo dedicado ao MS SQL Server 2012 edição Developer, vai ser abordada a segurança do SGBD, no âmbito do controlo de acesso à informação armazenada na base de dados AdventureWorks2012[12]. Serão analisados os mecanismos disponíveis para a implementação das políticas e modelos de controlo de acesso que estão disponíveis ao administrador do SGBD, que como foi descrito no capítulo anterior, é a autoridade central que gere um sistema de gestão de bases de dados. É a este que compete a atribuição e revogação de privilégios aos utilizadores do sistema, de acordo com a política implementada, assim como a responsabilidade da classificação dos utilizadores e dos dados armazenados no sistema [4, p. 838].

A conta do administrador é denominada de conta de sistema ou de superutilizador que faculta privilégios máximos que não estão disponíveis em mais nenhum tipo de conta de utilizador e cujas ações também já foram enumeradas, sendo a partir desta conta que é gerida a segurança do SGBD.

Existe atualmente uma enorme diversidade de SGBDs, para todos os sistemas operativos, com diferentes finalidades, sendo alguns deles gratuitos e outros que necessitam de licença, consoante também as suas edições. O MS SQL Server 2012 foi o SGBDs eleito para a realização do estudo, dado ser um dos SGBDs mais populares[13]. A escolha recaiu sobre versão 2012, que atualmente é a penúltima versão lançada pela Microsoft deste SGBD, sendo a última a versão 2014. A versão 2014 não foi por nós escolhida, no âmbito desta investigação, uma vez que à data em que este estudo foi iniciado esta versão ainda não tinha sido lançada. Dentro da versão MS SQL Server 2012 existem disponíveis as seguintes edições[14]:

- *Enterprise;*
- *Business Intelligence;*
- *Standard;*

[12] http://msftdbprodsamples.codeplex.com/releases/view/93587

[13] DB-Engines Ranking - http://db-engines.com/en/ranking

[14] Principal Editions of SQL Server 2012
http://msdn.microsoft.com/en-us/library/ms144275(v=sql.110).aspx

- *Web;*
- *Developer;*
- *Express;*

A edição utilizada no presente trabalho é a *Developer Edition*, uma vez que tem disponíveis todas as funcionalidades da *Enterprise Edition*, mas com licença gratuita para efeitos de testes.

É do lado do administrador do SGBD que vai ser realizado o presente estudo e que vai analisar o MS SQL Server 2012, através da realização de testes e na perspetiva das características referidas na secção 2.3.1.

Para a verificação das permissões que vão ser atribuídas aos diferentes utilizadores, as ações que vão ser testadas são as que constam no acrónimo CRUD[15] (Create, Read, Update, Delete) que descreve as ações básicas numa base de dados e que são realizadas através dos seguintes comandos SQL: INSERT, SELECT, UPDATE e DELETE.

As métricas NIST a utilizar visam avaliar o SGBD na implementação dos diferentes modelos de controlo de acesso e, na forma como gerem as permissões atribuídas aos diferentes utilizadores de acordo com os modelos a implementar. As funções contempladas nas métricas de avaliação vão então analisar através dos seus itens cada um dos pontos a observar no controlo de acesso gerido pelo SGBD ao nível das bases de dados.

[15] http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

3.1. Controlo de Acesso no MS SQL Server 2012

Antes de descrever o funcionamento do controlo de acesso no MS SQL Server 2012, é necessário conhecer alguns termos utilizados neste SGBD e que são descritos em [13, pp. 160 - 161].

É com base nestes conceitos que a gestão de acessos é efetuada, tanto ao nível do servidor como das bases de dados, sendo estes definidos por:

3.1.1. *Principals*

São entidades ou objetos que podem pedir acesso a objetos existentes no SGBD, existindo dentro destes, 3 tipos distintos e que são:

1. *Windows Principals* - São contas de utilizador ao nível local ou de domínio que são utilizadas para autenticação através do Windows.
2. *SQL Server Principals* - São SQL Server *logins* utilizados para autenticação no SQL Server através da sua própria autenticação.
3. *Database Principals* - São utilizadores, *users*, da base de dados, *roles* da base de dados ou *roles* das aplicações.

Na Figura 2 é mostrada a hierarquia de permissões no motor da base de dados, onde do lado esquerdo se encontram os *Principals*.

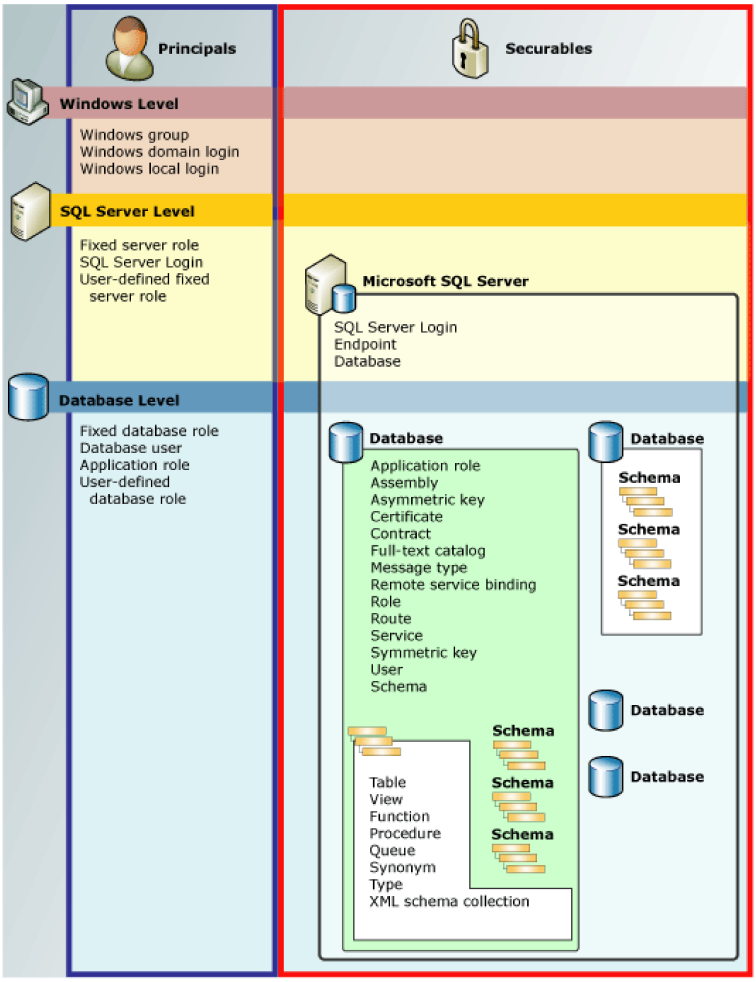

Figura 2 – Hierarquia de permissões no motor da base de dados[16].

3.1.2. *Securables*

São objetos do MS SQL Server e que podem ser geridos e controlados pela segurança do MS SQL Server, nomeadamente pelos *principals* de acordo com as permissões atribuídas a estes, Figura 2. Estes objetos podem ter três diferentes categorias e que são:

1. *Server Scope* - São objetos que se encontram ao nível do servidor, tais como *endpoints*, *logins*, *server roles* e bases de dados. Neste trabalho não serão tratados estes objetos.
2. *Database Scope* - Dentro desta categoria estão abragidos objetos tais como utilizadores, *roles*, *assemblies* e outros que não pertençam a *schemas* em específico.

[16] Fonte: https://msdn.microsoft.com/en-us/library/ms191465(v=sql.110).aspx

3. *Schema Scope* - Aqui são contemplados tipos, XML *Schemas* e objetos. Sendo que os objetos são tabelas, *views*, *stored procedures* e outros objetos criados dentro de *schemas*. Onde um *schema* é um grupo de objetos ao qual são atribuídos utilizadores, tendo estes utilizadores por sua vez permissões sobre os diferentes objetos contidos no *schema*, fica entre o nível da base de dados e dos objetos [14, p. 20].

Outro conceito importante no MS SQL Server é o de *owner*/dono/proprietário em algumas situações indicados como dbo (*database owner*). Um *database owner* [14, p. 18] é o dono de uma base de dados, ou qualquer outro utilizador que pertença à *server role* ***sysadmin***.

As características identificadas na secção 2.2 e 2.3 serão objeto de análise no contexto do MS SQL Server, como ilustram as secções seguintes.

3.1.3. *Logins* e Utilizadores

No MS SQL Server 2012 a gestão de controlo de acesso é realizada, sempre que possível através de ambiente gráfico, com recurso a uma ferramenta que é disponibilizada pela Microsoft para esse efeito. Esta ferramenta denominada de MS SQL Server Management Studio permite gerir as permissões de acesso, tanto ao servidor como às bases de dados nele contidas, entre outras funcionalidades.

De acordo com a ação referida na secção 2.3, relativamente à criação de conta, existem dois tipos de autenticação (*login*) no MS SQL Server, uma através da autenticação do Windows e outra com autenticação no MS SQL Server, mediante a introdução do nome de utilizador e respetiva *password*. Ao criar um *login* no servidor, pode-se optar pela autenticação Windows, ou por ambas, sendo selecionada a opção de *Mixed Mode*. De salientar que não é permitido selecionar unicamente autenticação MS SQL Server, Figura 3.

Figura 3 - Opções de autenticação no Microsoft SQL Server

O MS SQL Server possibilita aos *Windows Principals*, utilizadores locais do Windows, ou grupos de domínio e utilizadores, efetuarem a autenticação sem a necessidade de inserirem a sua password aquando da identificação. Através da autenticação Windows, que é recomendada sempre que possível, porque é mais segura [13, p. 162] os utilizadores autenticam-se no MS SQL Server através da sua conta no Windows. A autenticação com a conta do MS SQL Server implica ao utilizador introduzir o seu nome de utilizador e a respetiva *password*, ficando estas armazenadas numa base de dados.

Para a realização do estudo iremos criar cinco *logins* no servidor e cinco utilizadores na base de dados AdventureWorks2012. Desta forma, cada um dos cinco utilizadores vai ter um *login* associado para que possa autenticar-se no servidor e aceder à base de dados. De salientar que é necessário criar um *login* no servidor antes do respetivo utilizador na base de dados, de outra forma o utilizador ficará sem um *login* associado e é denominado utilizador orfão[17]. Para a autenticação de cada um dos novos *login*, é selecionada a autenticação através do MS SQL Server, uma vez que apesar da autenticação Windows ser mais segura, implicaria a criação de uma nova conta no Windows para cada novo *login*. Para a criação de um *login*, que é um objeto do âmbito do servidor, é no nó *Security* do servidor na opção *New Login*, como mostrado na Figura 4.

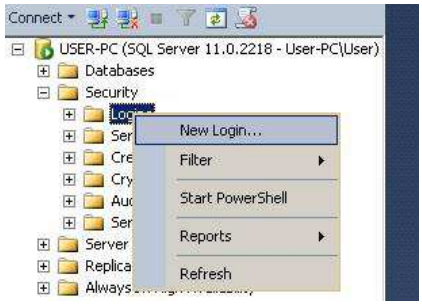

Figura 4 - Seleção da opção de criar um novo login no SQL Server 2012

Este novo *login*, terá a identificação de Login1 com *password* login1, com a base de dados AdventureWorks2012 como base de dados por defeito, Figura 5.

[17] https://msdn.microsoft.com/en-us/library/ms175475(v=sql.110).aspx

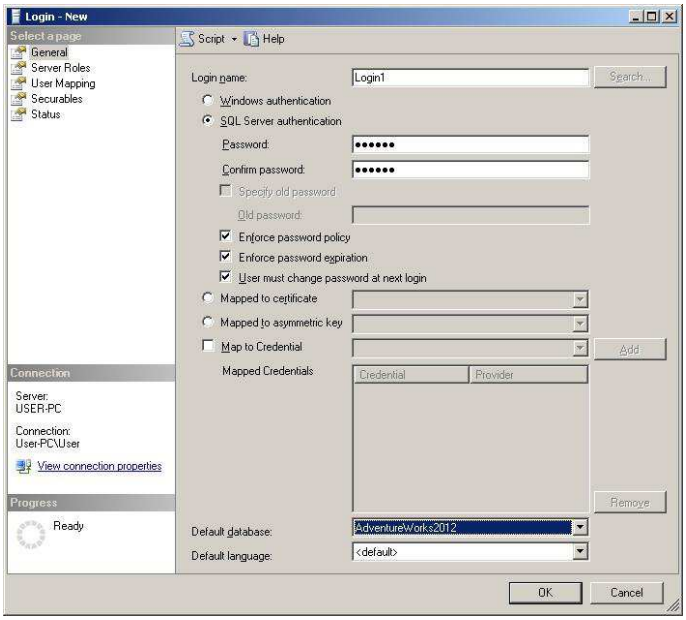

Figura 5 - Interface para a criação de um novo login no MS SQL Server 2012

Assim, ao ser criado um utilizador, *user*, este deve ficar logo associado a um *login* para que tenha acesso ao servidor SQL Server, por exemplo User1 na base de dados AdventureWorks2012, que vai ficar associado ao *login* Login1, Figura 6. Este utilizador não fica com nenhum *schema* atribuído, ficando essa ação para depois, consoante o modelo de controlo de acesso, requeira ou não a associação a um *schema*.

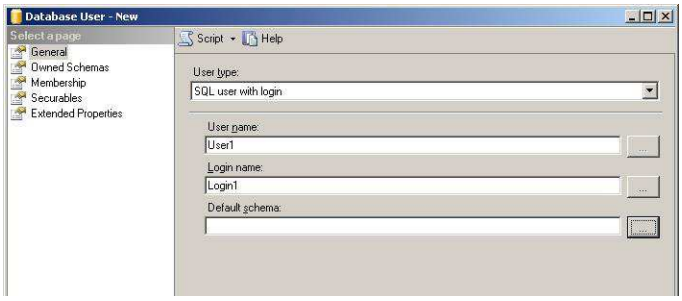

Figura 6 - Interface para a criação de um novo utilizador na base de dados

O User1 pode agora efetuar a sua autenticação com o Login1 e efetuar as ações para as quais tem permissões na base de dados onde foi criado. De salientar que as permissões que de início lhe estão atribuídas, são as que estão por defeito associadas à criação de um novo *login*. Isto é, um utilizador enquanto não lhe forem atribuídas permissões, simplesmente detém uma conta no sistema, mas não pode efetuar qualquer ação, pertencendo o seu login à *public server role*[18]. Esta *role* contém as permissões que são atribuídas por defeito a qualquer *login* e que podem ser parametrizadas pelo

[18] Server Level Roles - https://msdn.microsoft.com/en-us/library/ms188659(v=sql.110).aspx

administrador. A secção seguinte 3.1.4. mostra com profundidade os conceitos relacionado com as *roles* que serão aplicados no presente trabalho.

Para prosseguir com o estudo, vão ser criados mais 4 *logins* e 4 utilizadores para os ensaios de controlo de acesso, de forma a existirem 5 *logins* e 5 utilizadores. Estes estão identificados conforme a Tabela 1, onde consta a informação relativa aos utilizadores do SGBD.

Login	**Utilizador**	***Password***
Login1	User1	user1
Login2	User2	user2
Login3	User3	user3
Login4	User4	user4
Login5	User5	user5

Tabela 1 - Dados dos utilizadores do SGBD

Na Figura 7 são mostrados os 5 utilizadores criados na base de dados AdventureWorks2012, sendo que não estão atribuídas quaisquer permissões para além das definidas na *public database role*[19], uma vez que esta *role* (tal como a *public server role*) é atribuída a todos os utilizadores, mas para permissões ao nível da base de dados. Contém as permissões que são atribuídas por defeito a qualquer utilizador, podendo estas permissões ser definidas pelo administrador[20]. Estas *public roles* (*public server role* e *public database role*) não podem ser consideradas *roles* fixas, uma vez que as suas permissões são parametrizáveis, devendo existir especial cuidado uma vez que são atribuídas a qualquer utilizador.

Figura 7 - Utilizadores criados na BD AdventureWorks2012

[19] Database Level Roles - https://msdn.microsoft.com/en-us/library/ms189121(v=sql.110).aspx
[20] The public Role - https://msdn.microsoft.com/en-us/library/bb669065%28v=vs.110%29.aspx

3.1.4. *Database-Level Roles and Schemas*

No âmbito das ações passíveis de serem realizadas para garantir o controlo de acesso, existe a ação 2, de atribuição de privilégios. Neste contexto, nas bases de dados existem as *roles* que são *database principals* e que ajudam a gerir as permissões à base de dados. Dentro destas *roles* existem *roles* fixas e flexíveis, onde as fixas existem por defeito no SGBD e encontram-se disponíveis em todas as bases de dados, e as flexíveis são criadas e parametrizadas pelo utilizador. Na Tabela 2 estão as *roles* fixas que existem em qualquer base de dados assim como a sua descrição.

Role	**Descrição**
db_owner	Esta *role* permite aos seus membros executar todas as configurações e manutenções na base de dados, bem como eliminar a base de dados.
db_securityadmin	Os membros desta *role*, estão habilitados a modificar o conjunto de membros pertencentes à *role*, assim como definir permissões.
db_accessadmin	Os seus membros podem adicionar ou remover acessos à base de dados para Windows logins, grupos do Windows e SQL Server logins.
db_backupoperator	Os seus membros podem efetuar *backups* da base de dados.
db_ddladmin	Os seus membros podem correr qualquer comando DDL (Data Definition Language) na base de dados.
db_datawriter	Os seus membros podem adicionar, eliminar e alterar dados nas tabelas de todos os utilizadores da base de dados.
db_datareader	Os seus membros podem ler todos os dados nas tabelas de todos os utilizadores.
db_denydatawriter	Os seus membros não podem adicionar, modificar ou apagar dados nas tabelas dos utilizadores da base de dados.
db_denydatareader	Os seus membros não podem ler dados nas tabelas dos utilizadores da base de dados.

Tabela 2 - Lista de *database-level roles*[21] fixas

Além das *roles* fixas atrás mencionadas, ainda existe a *public database role*, também presente em todas as bases de dados a que pertencem todos os utilizadores aquando da sua criação, sendo esta *role* atribuída por defeito a todos os utilizadores. Esta *public database role* não consta desta tabela porque, como já referido, não é uma *role* fixa.

Segundo [15, p. 386], as *roles* são na maioria das situações a melhor prática para atribuir permissões aos *database principals*, nem que exista um único utilizador na *role*.

[21] http://msdn.microsoft.com/en-us/library/ms189121(v=sql.110).aspx

Na Figura 8 é mostrada a caixa de diálogo para a criação de uma *role* flexível, em que no campo superior é colocado o nome da *role*, no caso RoleTest, no campo abaixo a identificação do utilizador que é dono da *role*, User1. Depois temos os *schemas* e os membros que pertencem à *role*, no exemplo somente pertence o User2. De notar que esta *role* não foi utilizada para efeitos de testes, tendo sido criada somente a título de exemplo para este caso.

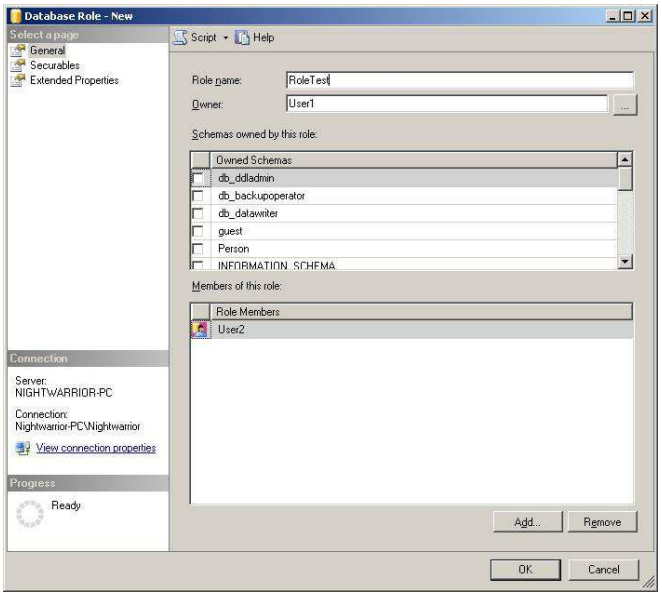

Figura 8 - Interface para criação de uma role flexível

Além das *roles*, existem os *schemas* que permitem agrupar objetos numa base de dados, permitindo uma gestão de segurança ao nível do grupo de objetos e facultando ao dono do *schema* uma maneira simples de atribuir permissões a outros utilizadores sobre os objetos pertencentes ao *schema* [15, p. 400]. Os *schemas* podem conter objetos tais como:

- Tabelas;
- *Views*;
- Funções;
- Procedimentos;

Os *schemas* não são criados por defeito como as *database roles*. Por exemplo, na base de dados AdventureWorks2012 existem *schemas* previamente criados para essa base de dados considerando as questões de segurança dos dados nela armazenados. Para a criação de um novo *schema* na base de dados AdventureWorks2012, basta selecionar a

opção *New Schema* na pasta *Security* da respetiva base de dados, tal como mostrado na Figura 9.

Figura 9 - Criação de um novo *schema*

Sempre que é criado um novo utilizador na base de dados, este pode ficar com um *default schema* atribuído, que pode ser qualquer um dos *schemas* existentes na base de dados. Desta forma, este utilizador sempre que crie um novo objeto, este objeto fica contido nesse *schema* e pertencente ao utilizador que o criou. Ficam assim acessíveis a este utilizador, um conjunto de objetos, para a execução de *queries* [13, p. 172].

3.1.5. Permissões

O controlo de acesso deve permitir a atribuição e revogação de permissões (ver ação 2 e 3 da secção 2.3), bem como a negação de permissões. Permissões (ou privilégios) são as autorizações atribuídas aos *principals* e que determinam as ações que são permitidas sobre os *securables*. As permissões aqui descritas aplicam-se somente a alguns *principals* e *securables* que estão ao nível da base de dados, onde os *securables* são as tabelas e o *views*. No MS SQL Server 2012 e ao nível da base de dados existem três tipos básicos de funções que podem ser executadas com as permissões, sendo estas funções as seguintes:

Grant - Atribuir permissões;

Revoke - Revogar permissões;

Deny - Negar permissões.

Em seguida apresentamos os tipos de permissões que podem ser atribuídas, negadas e revogadas ao nível da base de dados e que se aplicam a objetos como tabelas e *views*, uma vez que são estes os abordados no presente trabalho.

- *ALTER*: Permite modificar o objeto.

- *CONTROL*: Faculta ao *principal* todas as permissões sobre o *securable*.

- *DELETE*: Permite eliminar dados do objeto.

- *INSERT*: Permite inserir novos dados no objeto.

- *REFERENCES*: Permite fazer referência ao objeto, como aquando da criação de uma chave estrangeira numa tabela.

- *SELECT*: Permite executar *queries* ou ler dados dos objetos.

- *TAKE OWNERSHIP*: Permite a quem é dada esta permissão tomar a posse do objeto.

- *UPDATE*: Permite efetuar alterações nos dados dos objetos.

- *VIEW CHANGE TRACKING*: Permite aceder ao histórico de operações efetuadas no objeto.

- *VIEW DEFINITION*: Permite ao utilizador visualizar metadados dos objetos.

Nesta lista não constam todos os grupos, sendo que uma lista mais completa destes grupos está disponível na Microsoft Developer Network [22]. De salientar que nos testes seguintes somente serão manipuladas as permissões para as ações básicas numa base de dados e que são descritas pelo acrónimo CRUD.

[22] http://msdn.microsoft.com/en-us/library/ms191291.aspx#_permissions

4. Controlo de Acesso Discricionário

Neste capítulo vai ser implementado um modelo de controlo de acesso discricionário (DAC), no MS SQL Server 2012 ao longo de vários testes. Como já referido anteriormente na secção 2.2.1, este modelo visa um controlo de permissões sobre os objetos que é gerido pelos donos destes, sem a supervisão de uma entidade superior que normalmente é o administrador.

4.1. Controlo de Acesso Discricionário no MS SQL Server 2012

A implementação de um modelo de controlo de acesso discricionário no MS SQL Server 2012, visa atribuir permissões aos utilizadores sobre os objetos que lhes pertencem dentro de uma base de dados. Para a implementação deste modelo vai ser utilizada a base de dados AdventureWorks2012, uma vez que o modelo de controlo de acesso vai ser implementado ao nível da base de dados e não do servidor. A execução dos comandos será sempre feita do lado administrador, mas para testar as permissões dos diferentes utilizadores envolvidos nos testes vamos recorrer à personificação (*impersonated*) destes utilizadores através do comando EXECUTE AS[23].

Anteriormente foram dados a conhecer e descritos alguns mecanismos disponíveis no MS SQL Server 2012 para a gestão da segurança da base de dados. Através destes mecanismos, nomeadamente de *schemas*, que vai ser implementado e analisado o modelo discricionário de controlo de acesso.

Para tal vão fazer parte desta simulação 5 utilizadores criados na base de dados AdventureWorks2012, aos quais vão ser atribuídas permissões de maneira a que estes utilizadores possam (des)proteger os seus objetos, tendo sobre estes total controlo, bem como a possibilidade de atribuirem ou revogarem permissões sobre os seus objetos aos outros utilizadores, características obrigatórias de um modelo de controlo de acesso discricionário.

Para a simulação deste modelo de controlo de acesso, vão também ser utilizados os *schemas*, que também já foram descritos anteriormente, e que facilitam a gestão da

[23] https://msdn.microsoft.com/en-us/library/ms181362(v=sql.110).aspx

segurança da base de dados. Na base de dados utilizada já existem alguns *schemas*, como se pode verificar na Figura 10.

Figura 10 - *Schemas* existentes na AdventureWorks2012

Neste modelo a simular, cada um dos utilizadores criados vai ter os seus próprios objetos, sendo para isso necessário que o administrador faculte a estes permissões para criar ou controlar objetos já existentes.

Cada um destes utilizadores vai ter um *schema*, no qual vão estar contidos os objetos que lhe pertecem e os quais este pode controlar. A utilização de *schemas* visa facilitar o controlo dos privilégios atribuídos aos utilizadores sobre os objetos, uma vez que desta forma estes objetos ficam agrupados, sendo necessário unicamente atribuir privilégios sobre o *schema*.

Na base de dados AdventureWorks2012 existem 5 *schemas* criados, e que são[24]:

- *HumanResources;*
- *Person;*
- *Production;*
- *Purchasing;*
- *Sales.*

Em seguida, são descritos vários testes realizados e que visam comprovar a implementação de um modelo de controlo de acesso discricionário. Na Tabela 3 estão

[24] http://technet.microsoft.com/en-us/library/ms124894(v=sql.100).aspx

os testes a serem efetuados, onde estes são enumerados e descritos os seus objetivos. De salientar que os utilizadores envolvidos nos testes foram criados no capítulo 3.1.3.

Teste	Objetivo
Teste 1	Verificação de permissões por defeito.
Teste 2	Atribuição de permissões
Teste 3	Facilidade na atribuição de permissões
Teste 4	Verificar o alcance das permissões atribuídas
Teste 5	Atribuição de permissões por parte do dono do objeto
Teste 6	Atribuição de permissões entre utilizadores
Teste 7	Revogação de permissões
Teste 8	Prevalência das permissões
Teste 9	Negação de permissões
Teste 10	Resolução de conflitos de permissões
Teste 11	Atribuição de permissões de administração
Teste 12	Revogação de permissões de administração
Teste 13	Registo de ocorrências
Teste 14	Verificação de permissões atribuídas
Teste 15	Prevalência das permissões através da execução de *stored procedures*

Tabela 3 - Tabela com os testes realizados para o DAC

Teste 1 (Verificação de permissões por defeito): O objetivo é verificar as permissões que um utilizador tem por defeito na base de dados AdventureWorks2012 executando consultas (SELECT, INSERT, UPDATE).

Utilizadores envolvidos: User1.

Passos efetuados para a realização do teste:

1 - O User1 vai efetuar uma consulta de leitura (SELECT) numa tabela do *schema* Human Resources, através do seguinte *query*:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
SELECT * FROM AdventureWorks2012.HumanResources.Department;
REVERT;
```

Onde é devolvido o seguinte resultado:

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'Department', database
'AdventureWorks2012', schema 'HumanResources'.
```

Verifica-se que o User1 não tem permissões de leitura sobre essa tabela.

2 - O User1 vai também executar comandos de leitura (SELECT) em tabelas de outros *schemas* e que são as seguintes:

```
Person.Address;
Production.BillOfMaterials;
Sales.CountryRegionCurrency.
```

Verifica-se que o User1 não tem permissões de leitura sobre as tabelas associadas aos *schemas*.

3 - O User1 ao executar comandos para inserir (INSERT), alterar (UPDATE) e remover (DELETE) dados destas tabelas, verifica que são todos negados de igual forma.

Conclui-se que por defeito não há autorização para efetuar qualquer consulta à base de dados. A ausência de autorização é considerada como uma não autorização por parte do SGBD, implementando assim a política *closed world* [2, p. 6].

Teste 2 (Atribuição de permissões): O objetivo deste teste é atribuir permissão ao User1 para efetuar consultas de leitura (SELECT) na tabela Department. O responsável pela atribuição é o administrador, referido na secção 2.3.

Utilizadores envolvidos: User1

Passos efetuados para a realização do teste:

1 - Para autorizar o User1 a efetuar consultas na tabela Department, é possível atribuir somente essa permissão através do seguinte comando:

```
USE AdventureWorks2012
GO
GRANT SELECT ON AdventureWorks2012.HumanResources.Department TO User1;
GO
```

Ficando o User1 autorizado a efetuar consultas na tabela Department, conforme se verifica na Figura 11.

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2002-06-01 00:00:00.000
2	2	Tool Design	Research and Development	2002-06-01 00:00:00.000
3	3	Sales	Sales and Marketing	2002-06-01 00:00:00.000
4	4	Marketing	Sales and Marketing	2002-06-01 00:00:00.000
5	5	Purchasing	Inventory Management	2002-06-01 00:00:00.000
6	6	Research and Development	Research and Development	2002-06-01 00:00:00.000
7	7	Production	Manufacturing	2002-06-01 00:00:00.000
8	8	Production Control	Manufacturing	2002-06-01 00:00:00.000
9	9	Human Resources	Executive General and Administration	2002-06-01 00:00:00.000
10	10	Finance	Executive General and Administration	2002-06-01 00:00:00.000

Figura 11 - Resultado obtido da execução do *query* SELECT à tabela Department

2 - Também foram executadas consultas tais como inserir, alterar e remover dados. Contudo, como só foi atribuída permissão para SELECT, a execução destas consultas foi negada. Também foram efetuadas consultas sobre outras tabelas (Employee, JobCandidate), mas todas elas foram igualmente negadas.

Conclui-se assim que o administrador pode dar permissões aos utilizadores de forma independente sobre qualquer um dos objetos contidos na base de dados, sem afetar as permissões dos outros objetos.

Teste 3 (Facilidade na atribuição de permissões): O objetivo deste teste é verificar quais os passos necessários para atribuir, revogar e negar permissões a um utilizador. Como verificado no teste anterior, uma conta de utilizador ao ser criada não fica com qualquer permissão, tendo as permissões de ser atribuídas, ou pelo administrador do SGBD ou pelo dono do objeto sobre o qual vão ser dadas as permissões. De notar que as permissões têm de ser dadas para cada objeto e para cada tipo de consulta (SELECT,INSERT, etc.). Considerando estes passos verifica-se que não é fácil atribuir as permissões e as coisas complicam-se se houver muitos *securables* e utilizadores. Contudo, e como foi referido anteriormente, o SQL Server tem uma forma de agrupar objetos, os *schemas*, que permitem facilmente agrupar os objetos, para os atribuir a um utilizador (no caso do dono do *schema*), ou para atribuir permissões sobre os objetos a vários utilizadores. Neste teste, a permissão é dada pelo administrador aos utilizadores sobre os *schemas* que agrupam várias tabelas, conforme a Tabela 4, adquirindo cada um dos utilizadores a propriedade do *schema* que lhe é atribuído, ou seja fica como dono, desse *schema* e de todos os objetos nele contidos.

Utilizadores envolvidos: User1, User2, User3, User4 e User5.

Passos efetuados para a realização do teste:

1 - Atribuir cada um dos *schemas* aos utilizadores criados, ficando estes como sendo donos dos *schemas*, pela seguinte ordem:

User1 -> HumanResources;
User2 -> Person;
User3 -> Production;
User4 -> Purchasing;
User5 -> Sales.

Na tabela seguinte, os *schemas* estão dispostos pelas colunas, tendo as tabelas neles contidas, abaixo.

HumanResources	Person	Production	Purchasing	Sales
Department	Address	BillOfMaterials	ProductVendor	CountryRegionCurrency
Employee	AddressType	Culture	PurchaseOrderDetail	CreditCard
EmployeeDepartmentHistory	BusinessEntity	Document	PurchaseOrderHeader	Currency
EmployeePayHistory	BusinessEntityAddress	Illustration	ShipMethod	CurrencyRate
JobCandidate	BusinessEntityContact	Location	Vendor	Customer
Shift	ContactType	Product		PersonCreditCard
	CountryRegion	ProductCategory		SalesOrderDetail
	EmailAddress	ProductCostHistory		SalesOrderHeader
	Password	ProductDescription		SalesOrderHeaderSalesReason
	Person	ProductDocument		SalesPerson
	PersonPhone	ProductInventory		SalesPersonQuotaHistory
	PhoneNumberType	ProductListPriceHistory		SalesReason
	StateProvince	ProductModel		SalesTaxRate
		ProductModelIllustration		SalesTerritory
		ProductModelProductDescriptionCulture		SalesTerritoryHistory
		ProductPhoto		ShoppingCartItem
		ProductProductPhoto		SpecialOffer
		ProductReview		SpecialOfferProduct
		ProductSubcategory		Store

		ScrapReason		
		TransactionHistory		
		TransactionHistoryArchive		
		UnitMeasure		
		WorkOrder		
		WorkOrderRouting		

Tabela 4 - Tabelas contidas nos *schemas* da base de dados AdventureWorks2012

Na Figura 12 mostra-se a atribuição do *schema* HumanResources ao User1, sendo que para os restantes *schemas* e utilizadores o procedimento é idêntico. Como referido anteriormente, esta atribuição de permissões através de *schemas* visa facilitar a gestão das permissões.

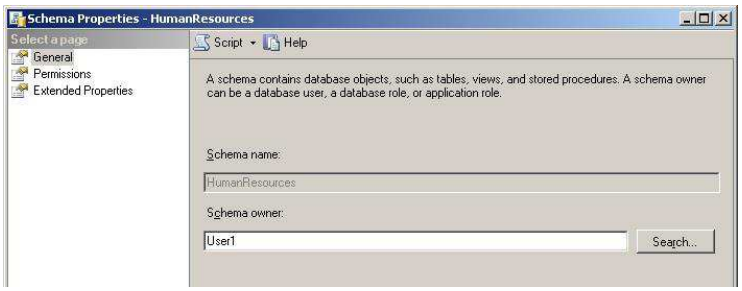

Figura 12 - Atribuição do schema HumanResources ao User1

2 - Verificação das permissões atribuídas com a seguinte consulta:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
SELECT * FROM AdventureWorks2012.HumanResources.JobCandidate;
GO
REVERT;
```

Assim, o User1 já tem permissão para efetuar SELECT sobre qualquer tabela pertencente ao *schema* HumanResources, sendo mostrado na Figura 13 o resultado da consulta realizada.

	JobCandidateID	BusinessEntityID	Resume	ModifiedDate
1	1	NULL	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2001-07-24 00:00:00.000
2	2	NULL	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2001-07-24 00:00:00.000
3	3	NULL	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2001-07-24 00:00:00.000
4	4	274	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2008-01-23 18:32:21.313
5	5	NULL	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2001-07-24 00:00:00.000
6	6	NULL	<ns:Resume xmlns:ns="http://schemas.microsoft.co...	2001-07-24 00:00:00.000

Figura 13 - Resultado do *query* à tabela JobCandidate pelo User1.

3 - Neste passo vai ser inserido um registo numa outra tabela do mesmo *schema* a fim de verificar se este utilizador também ficou com permissões de INSERT. Vai para isso ser utilizada a tabela EmployeePayHistory e o User1 vai inserir um registo através do seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
INSERT INTO AdventureWorks2012.HumanResources.EmployeePayHistory VALUES (4,
'2014-12-09', 25, 2, '2014-12-09');
GO
REVERT;
```

Que ao efetuar uma consulta à tabela pode-se verificar conforme a Figura 14, que este valor foi introduzido com sucesso.

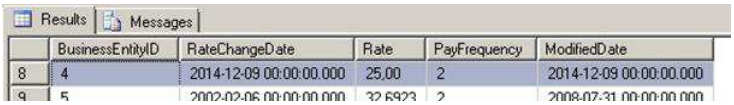

Figura 14 – Resultado da consulta à tabela EmployeePayHistory

4 - Verificar a permissão para alteração dos dados através do comando UPDATE a realizar pelo User1 conforme se segue:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
UPDATE AdventureWorks2012.HumanResources.EmployeePayHistory SET Rate=20 WHERE
RateChangeDate='2014-12-09';
GO
REVERT;
```

Verifica-se que o User1 tem permissão para efetuar alterações à tabela conforme se comprova na Figura 15.

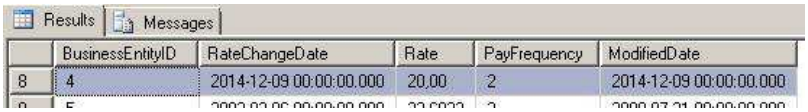

Figura 15 - Resultado da alteração a tabela EmployeePayHistory

5 – De igual forma foram realizados testes idênticos com os restantes utilizadores, nos seus respetivos *schemas*, verificando-se os mesmos resultados a nível de permissões.

Este teste permitiu mostrar os passos necessários para atribuir permissões aos utilizadores sobre as tabelas contidas nos diferentes *schemas*, ficando estes com autorização para consultar, inserir e alterar dados nas tabelas dos *schemas* que lhes pertencem.

Teste 4 (Verificar o alcance das permissões atribuídas): O objetivo deste teste é verificar se um utilizador tem permissões para criar uma tabela dentro de um *schema* do qual é dono.

Utilizadores envolvidos: User1.

Passos efetuados para a realização do teste:

1 - O User1 vai criar uma nova tabela dentro do *schema* HumanResources, com o comando seguinte:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
CREATE TABLE HumanResources.Table1 (IndexId int, Data nvarchar(50));
REVERT;
```

Não sendo no entanto permitido esta criação de tabela, apesar de ser "dono" deste *schema*, este utilizador ainda não tem permissões suficientes para tal, conforme se verifica pelo resultado devolvido.

```
Msg 262, Level 14, State 1, Line 1
CREATE TABLE permission denied in database 'AdventureWorks2012'.
```

2 - Para o utilizador ter autorização para criar uma tabela é necessário que o administrador do SGBD ou o dono da base de dados lhe conceda essa permissão, através do comando:

```
USE AdventureWorks2012
GRANT CREATE TABLE TO User1;
GO
```

Ficando o User1 com permissão para criar uma tabela dentro do *schema* HumanResources, para além daquelas que já tinha, ou seja, permissões para efetuar SELECT, UPDATE, INSERT e DELETE dos dados de tabelas do *schema*

HumanResources. De salientar que esta última permissão não se encontra ao nível das tabelas, mas sim já no nível da base de dados.

3 - O User1 ao tentar criar uma nova tabela dentro de outro *schema*, que não lhe pertença, como por exemplo o *schema* Person, com o comando que se segue:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
CREATE TABLE Person.Table2 (IndexId int, Data nvarchar(50));
GO
REVERT;
```

O resultado da execução é que o User1 não tem permissão para tal.

O User1 ao ser proprietário do *schema* HumanResources, não tem permissão para criar uma nova tabela dentro deste *schema*, uma vez que esta tarefa não se encontra ao nível do *schema* mas sim ao nível da base de dados. Estas permissões estão em níveis distintos.

Teste 5 (Atribuição de permissões por parte do dono do objeto): O objetivo deste teste é atribuir permissões a outros utilizadores através de um utilizador, sobre os objetos que pertencem a este último.

Utilizadores envolvidos: User1 e User2.

Passos efetuados para a realização do teste:

1 – Inserir dados na tabela criada no teste anterior, conforme o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
INSERT INTO AdventureWorks2012.HumanResources.Table1 VALUES (1, 'Data1');
GO
INSERT INTO AdventureWorks2012.HumanResources.Table1 VALUES (2, 'Data2');
GO
INSERT INTO AdventureWorks2012.HumanResources.Table1 VALUES (3, 'Data3');
GO
REVERT;
```

Este código foi executado com sucesso e os registos foram inseridos na tabela Table1.

2 - Efetuar uma consulta através de outro utilizador, que no caso vai ser o User2 com o comando que se segue,

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User2';
GO
SELECT * FROM AdventureWorks2012.HumanResources.Table_1;
GO
REVERT;
```

Obtendo-se o seguinte resultado:

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'Table1', database
'AdventureWorks2012', schema 'HumanResources'.
```

3 - O utilizador User1 vai conceder permissão ao User2 para consultar a tabela, executando o comando:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
GRANT SELECT ON AdventureWorks2012.HumanResources.Table1 TO User2;
GO
REVERT;
```

Na Figura 16 esta o diagrama de atribuição da permissão do User1 ao User2 sobre a tabela Table1 do *schema* HumanResources.

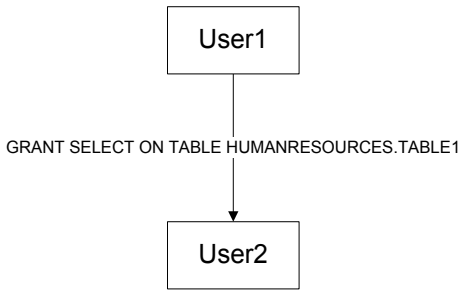

Figura 16 - Permissão de leitura do User1 ao User2 sobre a tabela Table1

Desta forma, o User2 já tem permissões para efetuar um SELECT à tabela Table1. A execução do SELECT foi realizada com sucesso como é mostrado na Figura 17.

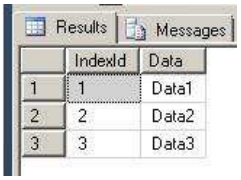

Figura 17 - Resultado do comando SELECT efetuado pelo User2.

Pode-se assim concluir que inicialmente o User2 não tinha permissões suficientes para efetuar uma consulta à tabela Table1 do *schema* HumanResources. Mas como definido no modelo de controlo de acesso DAC, o User1 sendo dono do *schema* HumanResources e de todos os objetos nele contidos, pode conceder permissões ao User2 para que este possa efetuar consultas nas tabelas contidas no *schema* HumanResources.

Este é um dos critérios do controlo de acesso discricionário, onde os utilizadores atribuem permissões sobre os seus objetos, sem que para isso seja necessário autorização superior do administrador.

Teste 6 (Atribuição de permissões entre utilizadores): O objetivo é verificar se um utilizador pode conceder as suas permissões a outros utilizadores, mesmo que ele não seja o dono, com a opção WITH GRANT.

Utilizadores envolvidos: User1, User2 e User3.

Passos efetuados para a realização do teste:

1 - O User2 com base no teste anterior vai então atribuir permissões de SELECT ao User3 sobre a tabela Table1, verificando que não pode efetuar esta operação, conforme o seguinte resultado:

```
Msg 15406, Level 16, State 1, Line 1
Cannot execute as the server principal because the principal "Login2" does not exist,
this type of principal cannot be impersonated, or you do not have permission.
```

2 - De seguida o User1 vai então conceder permissão ao User2 para que este possa conceder ao User3 a permissão para efetuar consultas na Table1, através do código WITH GRANT. O User1 tem de efetuar o comando que se segue:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
```

```
GRANT SELECT ON AdventureWorks2012.HumanResources.Table1 TO User2 WITH GRANT
OPTION;
GO
REVERT;
```

Sendo este código executado com sucesso.

3 - De seguida já é possível ao User2 conceder ao User3 permissão para efetuar uma consulta na tabela Table1, com o comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User2';
GO
GRANT SELECT ON AdventureWorks2012.HumanResources.Table1 TO User3;
GO
REVERT;
```

Na Figura 18 é mostrado o diagrama de atribuição final de permissões, sendo que o User1 é o *owner* do *schema* HumanResources onde se insere a tabela Table1.

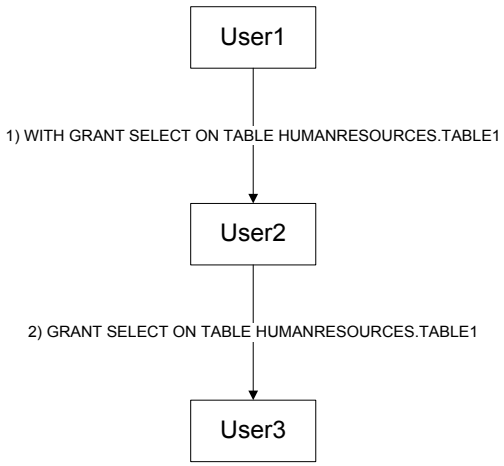

Figura 18 - Diagrama de atribuição de permissões na tabela Table1

Tal como mencionado no modelo de autorização Sistema R, os privilégios podem ser concedidos em cascata, tornando difícil o controlo dos objetos por parte do *owner* do objeto aquando da atribuição de permissões com a opção WITH GRANT.

Teste 7 (Revogação das permissões): O objetivo é revogar as permissões anteriormente atribuídas a outros utilizadores.

Utilizadores envolvidos: User1, User2 e User3.

Passos efetuados para a realização do teste:

1 - O User1 vai revogar o privilégio atribuído ao User2, sobre a tabela Table1, com o comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
REVOKE SELECT ON AdventureWorks2012.HumanResources.Table1 FROM User2;
GO
REVERT;
```

Obtendo o seguinte resultado:

```
Msg 4611, Level 16, State 1, Line 1
To revoke or deny grantable privileges, specify the CASCADE option.
```

Dado o User2 ter concedido permissões ao User3. Ou seja, como foi atribuída a permissão com WITH GRANT a revogação simples não foi possível. Teve que se fazer mais um passo (passo 2 a seguir) antes de revogar esta pemissão.

2 - A revogação desta permissão tem de passar pelo User1 revogar a opção WITH GRANT ao utilizador a quem se vai revogar a permissão atribuída. No caso é revogada a permissão WITH GRANT ao User2, através do comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
REVOKE GRANT OPTION FOR SELECT ON AdventureWorks2012.HumanResources.Table1 FROM
User2 CASCADE;
GO
REVERT;
```

O resultado da execução deste código é que o User3 ficará com a permissão atribuída pelo User2 revogada, sendo que o User2 ainda tem permissão para efetuar SELECT sobre a tabela.

3 - O User1 vai revogar a permissão de SELECT ao User2, executando o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
REVOKE SELECT ON AdventureWorks2012.HumanResources.Table1 FROM User2;
GO
REVERT;
```

A partir deste passo, tanto o User2 como o User3 viram os privilégios anteriormente concedidos, revogados.

Na Figura 19 é mostrado o diagrama de revogação de permissões.

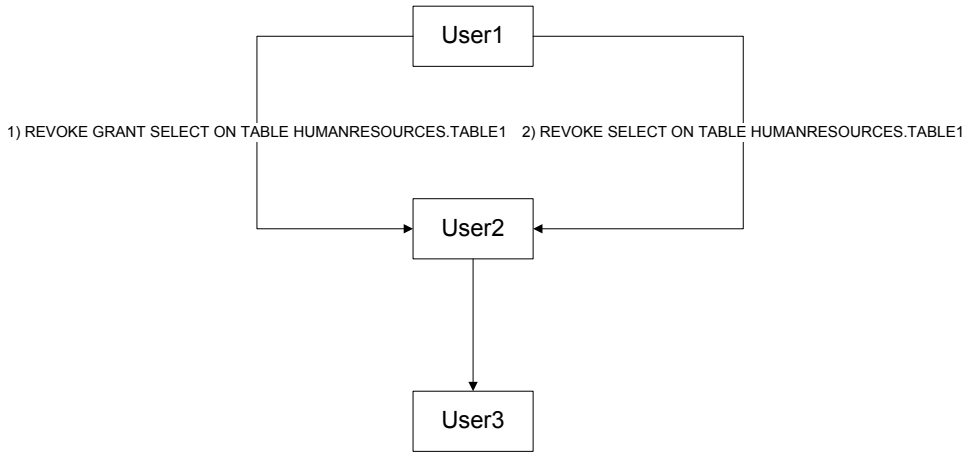

Figura 19 - Revogação de permissões do User1 ao User2

A mesma situação ocorre, mas de forma automática, se o administrador retirar o *schema* HumanResources ao User1, todas as permissões atribuídas por este utilizador a outros serão automaticamente revogadas. No exemplo do teste anterior, se o User1 deixar de ser dono do *schema* HumanResources, tanto o User2 como o User3 ficam automaticamente com as permissões, que lhes foram atribuídas e que tiveram origem no User1, revogadas.

Este SGBD não permite efetuar uma revogação de permissões *noncascading*, tal como se menciona anteriormente na secção dedicada ao modelo de autorização Sistema R, ou seja, não suporta a extensão referida na secção 2.2.1.

Teste 8 (Prevalência das permissões): O objetivo é verificar que a negação prevalece sobre a atribuição. Para tal iremos negar uma permissão de consulta de leitura a um utilizador sobre uma tabela de um *schema*, tendo este já a permissão de consulta de leitura sobre essa tabela do *schema.*

Utilizadores envolvidos: User2 e User3.

Passos efetuados para a realização do teste:

1 - Vai ser dada autorização ao User3 pelo User2 (primeira ação da Figura 20) para efetuar consultas nas tabelas contidas no *schema* que pertence ao User2, que é o Person, com o comando:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User2';
GO
GRANT SELECT ON SCHEMA::Person TO User3;
GO
REVERT;
```

Assim o User3 fica autorizado a efetuar consultas em todas as tabelas do *schema* Person.

Mas supondo que o User3 não pode efetuar consultas numa das tabelas, como por exemplo à tabela PersonPhone.

2 - O User2 vai então negar o acesso do User3 a essa tabela através do comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User2';
GO
DENY SELECT ON AdventureWorks2012.Person.PersonPhone TO User3;
GO
REVERT;
```

Na Figura 20 esta o diagrama de atribuição de permissões do User2 ao User3 para a tabela PersonPhone do *schema* Person.

Figura 20 - Diagrama de atribuição de permissões do User2 ao User3

Agora a permissão de consulta sobre o schema mantém-se, mas foi negada à tabela PersonPhone.

3 - O User3 tentar efetuar uma consulta à tabela PersonPhone com o comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User3';
GO
SELECT * FROM AdventureWorks2012.Person.PersonPhone;
GO
REVERT;
```

Obtendo o resultado que se segue:

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'PersonPhone', database
'AdventureWorks2012', schema 'Person'.
```

Verifica-se assim, para as situações testadas que a negação da autorização prevalece, bem como a implementação da exceção.

Teste 9 (Negação de permissões): O objetivo é atribuir uma permissão a um utilizador que tem essa permissão negada ao *schema* onde essa tabela se encontra. Também neste teste, se vai verificar o comportamento do SGBD no tratamento de exceções.

Utilizadores envolvidos: User2 e User4.

Passos efetuados para a realização do teste:

1 - Vai ser negada a permissão de efetuar SELECT sobre o *schema* Person ao User4 pelo User 2. No comando que se segue é negada esta autorização.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User2';
GO
DENY SELECT ON SCHEMA::Person TO User4;
GO
REVERT;
```

2 - Em seguida vai ser dada permissão ao User4 para efetuar consultas na tabela CountryRegion.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User2';
GO
GRANT SELECT ON AdventureWorks2012.Person.CountryRegion TO User4;
GO
REVERT;
```

Na Figura 21 é mostrado o diagrama de atribuição de permissões do User2 ao User4 sobre o *schema* Person e sobre a tabela CountryRegion.

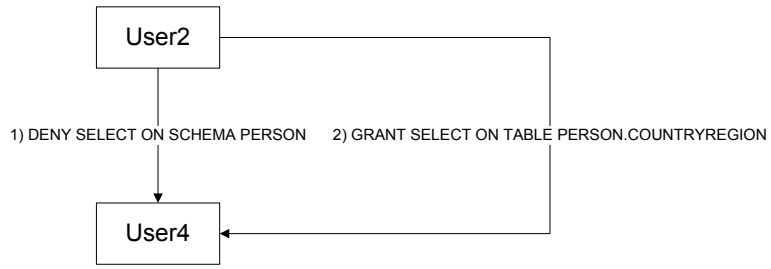

Figura 21 - Atribuição de permissões do User2 ao User4

3 - O User4 vai efetuar uma consulta a esta tabela e verifica que continua com a permissão negada, conforme se comprova com o resultado abaixo.

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'CountryRegion', database
'AdventureWorks2012', schema 'Person'.
```

Verifica-se assim que a negação prevalece, não sendo implementada a exceção.

Teste 10 (Resolução de conflitos de permissões): O objetivo é negar uma permissão a um utilizador por parte do dono do objeto, sendo que essa permissão vai ser dada por outro utilizador a quem o dono do objeto concedeu permissões de leitura sobre esse objeto.

Utilizadores envolvidos: User2, User3 e User4.

Passos efetuados para a realização do teste:

1 - O User2 vai atribuir permissões de leitura com opção WITH GRANT sobre a tabela Person do *schema* Person ao User3.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User2';
GO
GRANT SELECT ON AdventureWorks2012.Person.Person TO User3 WITH GRANT OPTION;
GO
REVERT;
```

Este comando foi executado com sucesso.

2 - De seguida o User3 vai atribuir permissões de leitura ao User4.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User3';
GO
GRANT SELECT ON AdventureWorks2012.Person.Person TO User4;
GO
REVERT;
```

Este comando foi executado com sucesso.

3 - O User2 vai negar o acesso de leitura ao User4 sobre a tabela Person.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User2';
GO
DENY SELECT ON AdventureWorks2012.Person.Person TO User4;
GO
REVERT;
```

O diagrama de atribuição de permissões fica conforme a Figura 22.

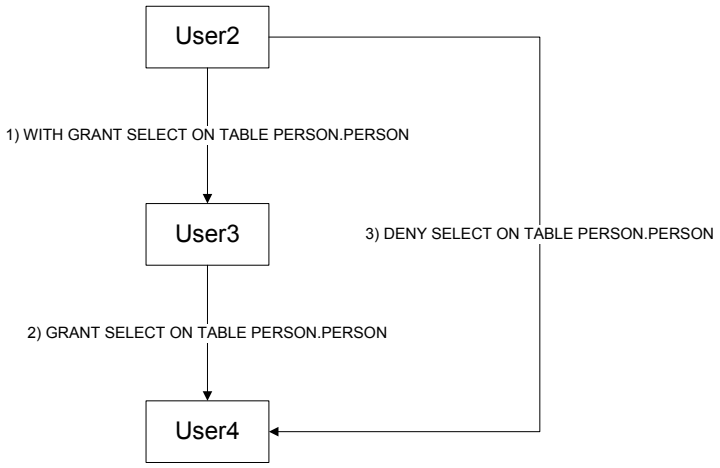

Figura 22 - Diagrama de atribuição de permissões

Este comando foi executado com sucesso.

4 - O User4 ao tentar efetuar uma consulta à tabela Person verifica que o seu pedido é negado.

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'Person', database
'AdventureWorks2012', schema 'Person'.
```

Este facto não ocorreu devido ao User3 não ter concedido a permissão, mas pelo facto do User2 lhe ter negado essa permissão, sendo o conflito de permissões resolvido com a prevalência da negação.

Teste 11 (Atribuição de permissões de administração): O objetivo deste teste é o dono de um *schema* atribuir permissões para controlo sobre o *schema* que lhe pertence, a outro utilizador. Para tal, um utilizador vai atribuir a permissão CONTROL a outro utilizador, onde por sua vez este último utilizador vai atribuir permissões a outro utilizador como dono do objeto.

Utilizadores envolvidos: User1, User3 e User5.

Passos efetuados para a realização do teste:

1 – O User1 vai atribuir permissão de controlo (CONTROL) ao User3 sobre o *schema* de que é dono (HumanResources). Para tal vai executar o comando que se segue:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
GRANT CONTROL ON SCHEMA::HumanResources TO User3;
GO
REVERT;
```

Este comando foi executado com sucesso.

2 – Em seguida o User3 vai efetuar uma consulta de leitura na tabela EmployeePayHistory com o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User3';
GO
SELECT * FROM AdventureWorks2012.HumanResources.EmployeePayHistory;
GO
REVERT;
```

E verifica que tem permissão para efetuar consultas de leitura na tabela. Foram também efetuadas outras consultas (INSERT, UPDATE, DELETE) a tabelas do *schema* HumanResources pelo User3 e verificou-se que tem permissões para tal.

3 – Como o User3 tem permissão de controlo sobre o *schema* HumanResources, vai por sua vez atribuir permissão de consultas de leitura e alteração ao User5, com o comando que segue:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User3';
GO
GRANT SELECT, UPDATE ON AdventureWorks2012.HumanResources.Employee TO User5;
GO
REVERT;
```

Tendo este comando sido executado com sucesso, verifica-se que o User5 tem permissões para efetuar consultas de leitura e alteração na tabela Employee.

Pode-se concluir através deste teste que é possível delegar privilégios de administração a outros utilizadores pelo dono do objeto sem autorização do administrador, tal como referido na secção 2.2.1 dedicada ao controlo de acesso discricionário.

Teste 12 (Revogação de permissões de administração): O objetivo deste teste é revogar as permissões de administração dadas no teste anterior.

Utilizadores envolvidos: User1, User3 e User5.

Passos efetuados para a realização do teste:

1 – O User1 vai revogar a permissão de controlo atribuída ao User 3 no teste anterior. Para tal vai executar o comando que se segue:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
REVOKE CONTROL ON SCHEMA::HumanResources TO User3;
GO
REVERT;
```

Tendo sido este comando executado com sucesso, o User3 já não dispõe de permissões de controlo sobre o *schema* HumanResources. Se o User3 tentar atribuir permissões de consulta de leitura a outro utilizador verifica que já não tem permissão.

2 – Mas o User3 tinha no teste anterior atribuído permissões de consultas de leitura e alteração na tabela Employee ao User5. Se o User5 tentar efetuar uma consulta de leitura nessa tabela, verifica que ainda tem permissão para a efetuar, como mostra a Figura 23, apesar de o utilizador que lhe atribuiu a permissão já ter a permissão revogada.

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User5';
GO
SELECT * FROM AdventureWorks2012.HumanResources.Employee;
GO
REVERT;
```

00 %

Results | Messages

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel
1	1	295847284	adventure-works\ken0	0x	0
2	2	245797967	adventure-works\terri0	0x58	1
3	3	509647174	adventure-works\roberto0	0x5AC0	2
4	4	112457891	adventure-works\rob0	0x5AD6	3
5	5	695256908	adventure-works\gail0	0x5ADA	3
6	6	998320692	adventure-works\josef0	0x5ADE	3

Figura 23 – Execução do consulta à tabela Employee pelo User5

Conclui-se deste teste que a revogação da permissão de controlo não revoga em cascata todas as permissões atribuídas a partir desta, efetuando assim uma revogação de permissões *noncascading*, como referido na secção 2.2.1 dedicada ao controlo de acesso discricionário.

Teste 13 (Registo de ocorrências): O objetivo é verificar o registo por parte do SGBD das ocorrências. Este teste é efetuado com recurso a uma ferramenta disponibilizada pelo MS SQL Server 2012 para auditoria, denominada de *Database Audit Specification*. Esta ferramenta permite ao administrador uma monitorização detalhada e registo, de todos os eventos que ocorrem dentro do servidor, podendo esta monitorização ser ao nível do servidor ou da base de dados. Para o presente trabalho só se vai tratar a auditoria ao nível da base de dados e a relação com o controlo de acesso. Vão ser usados dois utilizadores e ambos vão realizar operações na base de dados, sendo que só um deles vai ser monitorizado pelo administrador.

Utilizadores envolvidos: User1 e User2.

Passos efetuados para a realização do teste:

1 - Vai ser criado em primeiro lugar um *SQL Server Audit*, do lado do servidor, onde é definido o nome da auditoria (na ferramenta designada como *audit*), o tipo de saída de dados, que pode ser para ficheiro, *Windows Application Log*, ou *Windows Security Log*. Também é nesta caixa que se seleciona o destino do ficheiro, caso tenha sido esta a opção escolhida, Figura 24.

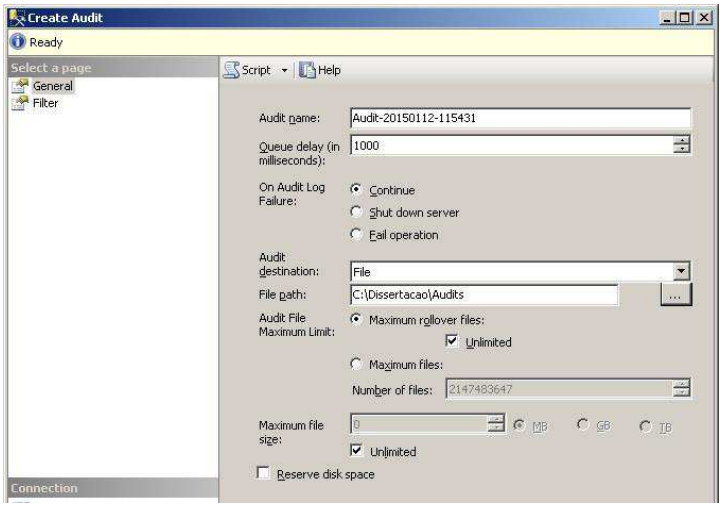

Figura 24 - Caixa para parametrizar a auditoria no servidor

2 - De seguida é necessário criar também a auditoria na base de dados. Para isso e como se mostra na Figura 25, dentro da árvore da base de dados e ao clicar com o botão do lado direito do rato surge um menu onde se seleciona a opção *New Database Audit Specification*.

Figura 25 - Criar uma nova Database Audit Specification

Em seguida, surge uma caixa para especificar a *audit* que vai ser criada, atribuindo-se um nome para a identificar, a ligação à auditoria criada anteriormente do lado do servidor, assim como o tipo de ação a ser monitorizada, a classe do objeto, o nome do objeto e o nome do *principal*.

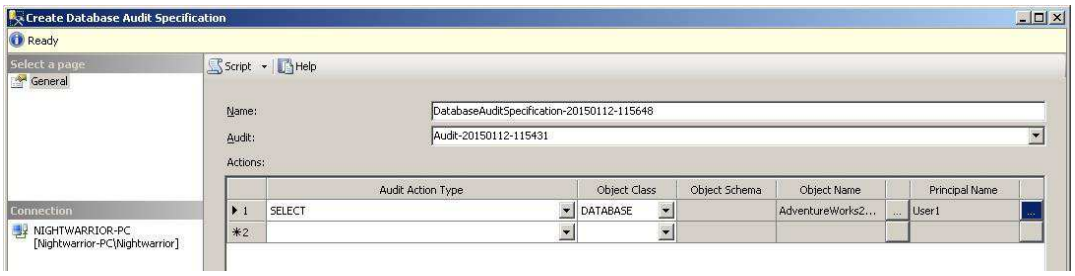

Figura 26 - Especificação da auditoria a ser criada na base de dados.

Como se mostra na Figura 26, para este teste o tipo de ação escolhido foi um SELECT sobre a base de dados AdventureWorks2012 pelo utilizador User1.

3 - Ao finalizar a configuração desta auditoria, é necessário ativar ambas as auditorias, tanto do lado do servidor como da base de dados. Para isso, é necessário escolher a opção *Enable* no menu que aparece quando se clica com o botão direito do rato, sobre as auditorias. Na Figura 27 mostra-se a ativação da *Database Audit Specification*.

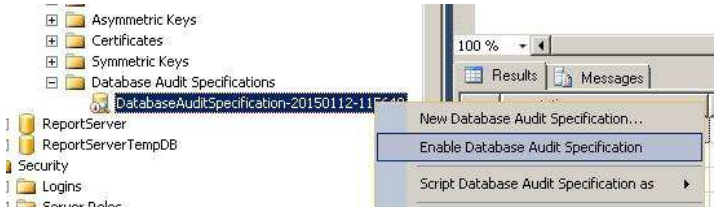

Figura 27 - Ativação da auditoria na base de dados

Na Figura 28 encontra-se a ativação da auditoria do lado do servidor.

Figura 28 - Ativação da auditoria do lado do servidor

Ao serem ativadas com sucesso, surge no final do processo de ativação uma caixa onde é dada a confirmação da ativação das auditorias. A Figura 29 mostra a caixa da ativação da auditoria do servidor.

Figura 29 - Confirmação da ativação da audit no servidor

A partir deste momento as duas auditorias já se encontram ativadas.

4 - O User1 vai efetuar dois comandos de SELECT, um sobre uma tabela para a qual tem permissão para efetuar esse comando e outro comando sobre outra tabela que não tem permissão de consulta.

5 - O User2 também vai efetuar um comando de SELECT sobre uma tabela em que tem permissão para tal e em outra que não tem. O objetivo é verificar que estas ações não

vão ficar registadas na auditoria, uma vez que este utilizador não está a ser monitorizado.

6 - O administrador vai agora consultar todos os eventos ocorridos e registados pelas auditorias, e que no caso mostra os eventos de todas as auditorias contidas na pasta "C:\Dissertacao\Audits".

```
SELECT * FROM fn_get_audit_file('C:\Dissertacao\Audits\*', default, default)
```

Se o administrador desejar consultar somente os eventos registados por uma auditoria, deve especificar o nome desta, para só serem mostrados esses resultados. O nome da auditoria pode ser visualizado dentro da pasta dada como destino para colocação das auditorias. Este nome, que é atribuído por defeito, é uma concatenação da data e hora a que foi criada a auditoria, porém pode ser dado outro nome à escolha.

7 - Neste passo o administrador vai efetuar uma consulta somente à auditoria agora criada e não a todas as auditorias existentes dentro da pasta.

```
SELECT * FROM fn_get_audit_file('C:\Dissertacao\Audits\Audit-20150112-
115431_1A0C2C80-B91A-4B5C-B9AE-8346434254F8_0_130655378333440000.sqlaudit',
default, default)
```

Na Figura 30 é mostrado o resultado compactado da consulta.

	event_time	s...	action_id	succeeded	permis...	is_...	se...	serv...	dat...	ta...	tar...	obj...	cla...	sessio...	server_pri...
1	2015-01-12 12:03:53....	0	AUSC	1	0x00...	0	52	259	0	0	0	0	A	Night...	Nightwar...
2	2015-01-12 12:12:29....	1	SL	1	0x00...	1	54	267	5	0	0	10...	U	Night...	Login1
3	2015-01-12 12:12:51....	1	SL	0	0x00...	1	54	267	5	0	0	37...	U	Night...	Login1

	datab...			database_name	schema_name	object_name	statement
1							
2	User1			AdventureWorks2012	HumanResources	Department	SELECT * FROM AdventureWorks2012.HumanResources.Department;
3	User1			AdventureWorks2012	Person	Address	SELECT * FROM AdventureWorks2012.Person.Address;

Figura 30 - Resultado da consulta à auditoria

Na tabela que resulta da consulta, cada um dos eventos registados pela auditoria fica armazenado por linha, neste caso apenas foram registados três eventos, estando os dados dos eventos divididos por colunas, as quais correspondem a diversa informação[25]. Podem-se destacar, no caso dos dois eventos detetados e que estão na linha 2 e 3, a ação efetuada (ver coluna *action_id*)[26] onde consta um SL (SELECT), por quem foi realizada (ver coluna *database_principal*), User1, uma vez que a auditoria só monitoriza este utilizador, e se a ação foi autorizada (coluna *Succeeded*). A primeira linha da tabela está

[25] Consultar o resultado em: http://msdn.microsoft.com/en-us/library/cc280545(v=sql.110).aspx

[26] Consultar os tipos de ações em: http://msdn.microsoft.com/en-us/library/cc280663(v=sql.110).aspx

relacionada com as características da auditoria, onde tem o registo "action_id = AUSC"[27]. Como foi referido anteriormente o User1 tinha permissões para fazer a primeira consulta, mas não tinha para fazer a segunda, isto é registado na auditoria na coluna *succeded*, que está com valor 1 para a primeira consulta e 0 para a segunda. Como se pode verificar na Figura 30, não foi registado qualquer evento do User2.

Outra ação que pode ser monitorizada nestas auditorias é a atribuição de permissões sobre objetos contidos em *schemas*.

8 - Neste passo é criada uma nova auditoria, tanto no servidor como na base de dados, que vai registar eventos que estão relacionados com a alteração de permissões em objetos de *schemas*, ver Figura 31.

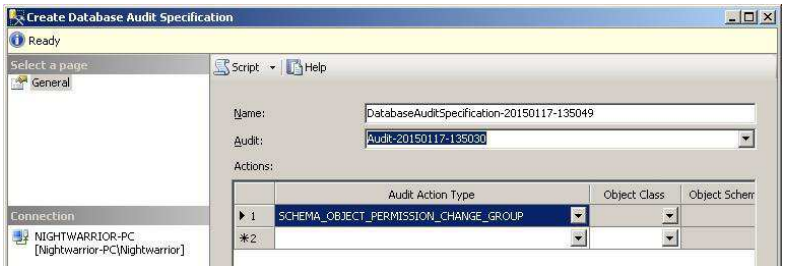

Figura 31 – Configuração da nova auditoria

Com as auditorias ativas, ao serem realizadas as atribuições de permissões descritas no teste 10, o resultado é o mostrado na Figura 32. Sendo esta, outra forma de ver o conteúdo de uma auditoria realizada. Aqui o resultado é mostrado através da opção *View Audit Logs* que se encontra no menu que aparece ao clicar com o botão direito do rato sobre a auditoria do lado do servidor. Nesta opção é mostrada uma caixa, que no lado esquerdo, permite selecionar qualquer uma das auditorias realizadas e, no lado direito, é apresentada uma tabela com o registo dos eventos.

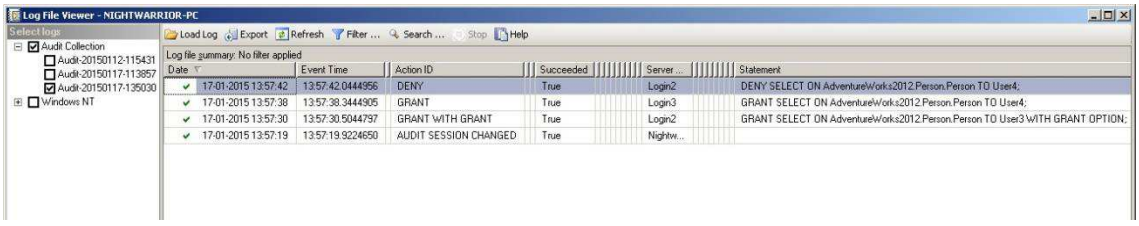

Figura 32 – Visualização dos registos da auditoria

[27] https://cprovolt.wordpress.com/2013/08/02/sql-server-audit-action_id-list/

Teste 14 (Verificação de permissões atribuídas): O objetivo é verificar que permissões se encontram atribuídas aos diferentes utilizadores da base de dados. Este teste tem como finalidade a visualização das permissões (*Grant* e *WithGrant*) atribuídas aos utilizadores sobre os objetos da base de dados. Para este fim, o SGBD não dispõe de uma ferramenta específica, tendo esta tarefa de ser efetuada com outros meios (consultas a tabelas). Para isso, existe uma tabela no SGBD que pode ser consultada e que tem todas as permissões atribuídas aos *principals* sobre os *securables*, esta tabela é a *fn_my_permissions*[28]. Assim, qualquer utilizador pode efetuar um *query* sobre esta tabela para ver quais as permissões que detém sobre um determinado objeto e colunas que o objeto possa ter.

Utilizadores envolvidos: User1, User2 e User4.

Passos efetuados para a realização do teste:

1 - O User1 vai efetuar um *query* sobre a tabela *fn_my_permissions* para ver quais as permissões que detém sobre a tabela Department do *schema* HumanResources.

```
USE AdventureWorks2012
GO
EXECUTE AS USER = 'User1';
GO
SELECT * FROM fn_my_permissions('AdventureWorks2012.HumanResources.Department',
'OBJECT');
GO
REVERT;
```

O resultado do *query* é mostrado na Figura 33 onde se mostram todas as permissões que o User1 tem sobre objeto (*entity_name*), as colunas contidas no objeto, no caso de existirem (*subentity_name*) e as ações que pode efetuar (*permission_name*) sobre o objeto.

[28] https://msdn.microsoft.com/en-us/library/ms176097(v=sql.110).aspx

	entity_name	subentity_name	permission_name
1	AdventureWorks2012.HumanResources.Department		SELECT
2	AdventureWorks2012.HumanResources.Department		UPDATE
3	AdventureWorks2012.HumanResources.Department		REFERENCES
4	AdventureWorks2012.HumanResources.Department		INSERT
5	AdventureWorks2012.HumanResources.Department		DELETE
6	AdventureWorks2012.HumanResources.Department		VIEW CHANGE TRACKING
7	AdventureWorks2012.HumanResources.Department		VIEW DEFINITION
8	AdventureWorks2012.HumanResources.Department		ALTER
9	AdventureWorks2012.HumanResources.Department		TAKE OWNERSHIP
10	AdventureWorks2012.HumanResources.Department		CONTROL
11	AdventureWorks2012.HumanResources.Department	DepartmentID	SELECT
12	AdventureWorks2012.HumanResources.Department	Name	SELECT
13	AdventureWorks2012.HumanResources.Department	GroupName	SELECT
14	AdventureWorks2012.HumanResources.Department	ModifiedDate	SELECT
15	AdventureWorks2012.HumanResources.Department	DepartmentID	UPDATE
16	AdventureWorks2012.HumanResources.Department	Name	UPDATE
17	AdventureWorks2012.HumanResources.Department	GroupName	UPDATE
18	AdventureWorks2012.HumanResources.Department	ModifiedDate	UPDATE
19	AdventureWorks2012.HumanResources.Department	DepartmentID	REFERENCES
20	AdventureWorks2012.HumanResources.Department	Name	REFERENCES
21	AdventureWorks2012.HumanResources.Department	GroupName	REFERENCES
22	AdventureWorks2012.HumanResources.Department	ModifiedDate	REFERENCES

Figura 33 - Permissões do User1 sobre a tabela Department

2 - O administrador, através do comando que se segue, ao executar como um qualquer utilizador, pode ver as permissões de todos os utilizadores da base de dados. Ao efetuar uma consulta para visualizar as permissões que o User 4 tem sobre a tabela Person, com o comando seguinte:

```
USE AdventureWorks2012
GO
EXECUTE AS USER = 'User4';
GO
SELECT * FROM fn_my_permissions('AdventureWorks2012.Person.Person', 'OBJECT');
GO
REVERT;
```

O resultado devolvido é mostrado na Figura 34, devendo-se salientar que esta permissão tinha sido negada ao User4 pelo User2 no teste 10. Como já foi demonstrado anteriormente, a negação sobrepõe-se à atribuição das permissões. É como se este utilizador não tivesse permissões atribuídas, razão pela qual não aparece qualquer permissão do User4.

	entity_name	subentity_name	permission_name

Figura 34 - Permissões que o User4 tem sobre a tabela Person

3 - Existe também uma *stored procedure* que permite visualizar as permissões sobre um determinado objeto da base dados, a *sp_helprotect*[29]. Esta *stored procedure* mostra as

[29] http://msdn.microsoft.com/en-us/library/ms190310(v=sql.110).aspx

permissões sobre o objeto e por quem foram dadas. O administrador ao executar o seguinte comando:

```
USE AdventureWorks2012;
GO
EXEC sp_helprotect @name=Person
```

Fica com a informação de quem tem (*Grantee*) permissões sobre a tabela Person e por quem (*Grantor*) foram concedidas, o tipo de proteção (*ProtectType*), a ação/*query* (*Action*) e colunas afetadas (*column*), Figura 35.

	Owner	Object	Grantee	Grantor	ProtectType	Action	Column
1	Person	Person	User3	User2	Grant_WGO	Select	(All+New)
2	Person	Person	User4	User2	Deny	Select	(All+New)
3	Person	Person	User4	User3	Grant_WGO	Select	(All+New)

Figura 35 - Resultado da execução da sp_helprotect sobre a tabela Person

Aqui se verifica mais uma vez que o User4 não tem permissões de consulta sobre a dita tabela, uma vez que tem essa permissão negada por parte do User2, que é o dono deste objeto (ver teste 10).

4 – Existe ainda mais uma tabela que contém permissões atribuídas, a tabela sys.database_permissions[30]. O administrador pode efetuar uma consulta a esta tabela para ver quais as permissões que o User1 tem, e verificar que não constam quaiquer permissões deste utilizador nesta tabela, conforme Figura 36.

```
SELECT * FROM sys.database_permissions AS DP WHERE DP.grantee_principal_id IN (USER_ID('User1'));
```

00 %

Results | Messages

class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc

Figura 36 – Consulta à tabela sys.database_permissions

Esta consulta de leitura à tabela sys.database_permissions sobre o User1 não devolve nenhum resultado, apesar de este utilizador ser dono do *schema* HumanResources, tal como verificado no passo 1 deste teste.

Conclui-se com este teste que o administrador não dispõe de uma ferramenta simples e rápida para identificar as permissões que cada um dos utilizadores tem na base de dados.

[30] https://msdn.microsoft.com/en-us/library/ms188367(v=sql.110).aspx

Teste 15 (Prevalência das permissões através da execução de *stored procedures*): Este teste tem como finalidade um utilizador efetuar uma ação para a qual não tem permissão, através de uma *stored procedure*. O User4 vai efetuar consultas à tabela EmployeePayHistory, não estando esta ação inicialmente atribuída, porque não foi concedida essa permissão (como se pode verificar no teste anterior, este utilizador não tem permissões atribuídas), ou no caso seguinte, essa permissão negada.

Utilizadores envolvidos: User4

Passos efetuados para a realização do teste:

1 - Vai ser criada uma *stored procedure* para efetuar consultas à tabela EmployeePayHistory, como se pode ver em seguida.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE sp_selectEmployeePayHistory
AS
BEGIN
	SET NOCOUNT ON;
	SELECT * FROM [HumanResources].[EmployeePayHistory]
END
GO
```

2 - Ao User4 vai ser dada permissão de execução ao *stored procedure* acima definido.

3 - O User4 vai executar a *stored procedure* e efetuar uma consulta à tabela EmployeePayHistory.

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User4'
GO
EXEC sp_selectEmployeePayHistory
GO
REVERT;
```

E assim se verifica que apesar de não ter permissão atribuída para efetuar uma consulta na tabela, este utilizador consegue realizar esta ação através da *stored procedure*.

4 - Ao User4 vai ser negada a permissão de execução ao *stored procedure* acima definido.

5 - O User4 executa novamente a *stored procedure* e a execuação é feita com sucesso, permitindo mais uma vez fazer o SELECT da tabela EmployeePayHistory, mesmo tendo essa permissão negada. De notar que não consegue fazer o SELECT diretamente à tabela, mas através da *stored procedure* consegue obter o mesmo resultado.

Através das *stored procedures* é possível permitir que os utilizadores efetuem ações sobre os objetos, neste caso à tabela EmplyeePayHistory, sem que lhe tenham sido atribuídas permissões, ou mesmo que estas permissões se encontrem negadas. Os utilizadores podem mesmo não ter conhecimento dos comandos que são executados dentro das *stored procedures* [15, p. 402], no entanto podem executá-los sem quaisquer restrições.

Face aos testes aqui realizados podemos afirmar que o MS SQL Server assenta, no contexto do controlo de acesso discricionário, no Sistema R, permitindo a atribuição e revogação de permissões. Também foi possível verificar que o SGBD suporta a revogação recursiva de autorizações a utilizadores que obtiveram estas de um utilizador que viu a sua autorização revogada, bem como a política *closed world*. Exceção para o caso das permissões de administração, em que as permissões atribuídas com base nestas não têm revogação recursiva. Cabe ainda destacar o papel dos *schemas*, como facilitador da gestão das permissões, bem como o papel das *stored procedures*, que pode ser visto como uma forma de contornar permissões revogadas ou negadas.

5. Controlo de Acesso Não Discricionário

Neste capítulo vão ser analisados os modelos considerados não discricionários mais conhecidos, de acordo com o NIST [3, p. 6], que são os modelos MAC e RBAC.

5.1. Controlo de Acesso Mandatório no MS SQL Server 2012

A implementação de um controlo de acesso mandatório do tipo MAC MLS, sendo este o mais conhecido e utilizado, requer a possibilidade de utilizar multiníveis no SGBD. Como descrito anteriormente na secção 2.2.2, este controlo de acesso impõe uma entidade superior que controla todo o tipo de acesso ao SGBD e às bases de dados. O presente trabalho incide apenas ao nível das bases de dados e não do servidor, sendo somente abordadas as possibilidades para o controlo de acesso neste nível.

Assim e como já referido, este modelo MAC MLS coloca todos os objetos e utilizadores em níveis de segurança, o que implica no caso do MS SQL Server 2012 a atribuição de níveis de segurança aos *Principals* e *Securables*, uma vez que a atribuição, revogação e restrição de permissões é feita através de *labels* com indicação do nível de segurança em que se encontram todos os *Principals* e *Securables*.

Uma vez que este SGBD não dispõe nativamente de ferramentas que suportem este modelo de controlo de acesso, existe no entanto um projeto denominado de *SQL Server Label Security Toolkit*[31] que faculta ferramentas e técnicas para implementar no MS SQL Server, nas versões 2005 a 2012, *row-level security* (RLS) e *cell-level security* (CLS) baseados em *security labels*. Este projeto é mesmo recomendado em [14] como solução para a implementação deste tipo de controlo de segurança.

Havia autores, que nos tempos primordiais do modelo RBAC, mencionavam a possibilidade de implementar modelos MAC com recurso a mecanismos que faziam uso de *roles*, como é descrito em [16].

[31] https://sqlserverlst.codeplex.com/

Também em [17], se mencionava a implementação de um modelo MAC através do uso de um sistema de controlo de acesso RBAC, ainda que o SGBD utilizado não seja o MS SQL Server.

Segundo [18], as implementações de modelos MAC em bases de dados relacionais e com MLS estão restritos a algumas organizações governamentais, existindo três SGBDs relacionais em comercialização e que suportam este modelo. Estes SGBDs são o Trusted Oracle, Informix Online/Secure e o Sybase Secure SQL Server, não sendo mencionado neste artigo qualquer referência à possível implementação de um modelo MAC sobre o MSSQL Server. Também outro SGBD dedicado à implementação de controlo de acesso MAC é o Trusted Rubix[32], que é, tal como os outros três, mais indicado para sistemas onde é exigida alta segurança.

Assim verifica-se que o MS SQL Server não vem preparado para implementar o modelo de controlo de acesso MAC, sendo necessário instalar outras ferramentas para que tal seja possível. Também devido ao facto de serem poucas as organizações que, nos dias de hoje, utilizam modelos de controlo de acesso MAC, este tipo de segurança não vai ser implementado no presente trabalho. De salientar que caso seja necessário a implementação de um modelo de controlo de acesso MAC, existem como atrás mencionado, SGBDs que suportam este tipo de controlo.

5.2. RBAC no MS SQL Server 2012

De acordo com [14], este será o modelo de controlo de acesso mais adequado para implementar no MS SQL Server 2012, sendo recomendado como boas práticas de segurança a utilização de *roles* para atribuição de permissões aos utilizadores. O presente trabalho somente trata de *roles* ao nível das bases de dados e não ao nível do servidor.

O MS SQL Server 2012 tem por defeito 9 *roles* fixas disponíveis em cada base de dados e já descritas anteriormente na Tabela 2. Também existe mais uma *database level role*, denominada de *public*, à qual qualquer novo utilizador pertence por defeito e que já foi tratada anteriormente no capítulo 3.1.3.

[32] http://rubix.com/cms/

Na Figura 37 estão as *roles* que pertencem à base de dados AdventureWorks2012.

Figura 37 - *Database level roles* da AdventureWorks2012

Dependendo das organizações onde é implementado este modelo de controlo de acesso, pode ser necessário criar novas *roles* de acordo com as funções e privilégios dos utilizadores da base de dados.

Nos testes que vão ser realizados, vão ser utilizados os cinco utilizadores previamente criados e que já foram alvo dos testes anteriores. Estes utilizadores vão deixar de ser donos dos *schemas* que lhes tinham sido atribuídos anteriormente, sendo as permissões unicamente atribuídas através das *roles*.

Serão criados cenários de atribuição e revogação de permissões, bem como de negação entre os diversos utilizadores, seguindo a sequência de testes a mesma ordem da efetuada nos testes de implementação do modelo DAC.

Todas as permissões atribuídas aos utilizadores nos testes anteriores vão ser removidas, passando os *schemas* existentes novamente para a posse do *database owner* (dbo).

Neste capítulo não vão ser testados os privilégios por defeito dos utilizadores, dado já ter sido efetuado durante os testes relativos ao DAC e onde se pôde concluir que o SGBD implementava a política *closed world*, em que a ausência de autorização pressupõe a sua negação. No entanto, e para confirmar a ausência das permissões é efetuada uma tentativa de consulta por parte de cada um dos utilizadores (User1 ao User5) às tabelas que se encontram nos *schemas* que lhes pertenciam. Uma vez verificada a ausência de permissões, pode então ser dado início à atribuição de permissões com recurso a *roles*.

Neste capítulo, não vão ser descritas as *roles* fixas, uma vez que esta descrição já foi apresentada no capítulo 3.1.4.

Na Tabela 5, estão enumerados os testes que vão ser realizados para a implementação de um modelo RBAC e os respetivos objetivos, os utilizadores envolvidos nestes testes são os mesmos que foram utilizados na secção 4.1 dedicada ao modelo discricionário, tendo estes sido criados na secção 3.1.3.

Teste	Objetivo
Teste 1	Atribuição de permissões
Teste 2	Facilidade na atribuição de permissões
Teste 3	Verificar o alcance das permissões
Teste 4	Atribuição de permissões entre utilizadores
Teste 5	Revogação das permissões
Teste 6	Prevalência das permissões atribuídas por *roles*
Teste 7	Resolução de conflitos de permissões
Teste 8	Atribuição de permissões de administração
Teste 9	Revogação de permissões de administração
Teste 10	Registo de ocorrências
Teste 11	Verificação de permissões atribuídas
Teste 12	Prevalência de permissões através da execução de *stored procedures*
Teste 13	Verificação de ações permitidas por *stored procedures*

Tabela 5 - Tabela com os testes realizados para o RBAC

Tal como nos testes realizados no capítulo anterior dedicado ao controlo discricionário, também aqui a execução dos comandos será feita do lado administrador, sendo as permissões dos utilizadores, testadas com recurso à personificação destes utilizadores através do comando EXECUTE AS[33].

Teste 1 (Atribuição de permissões): O objetivo deste teste é atribuir através de *roles*, permissão a um utilizador para consultar a tabela Department.

Utilizadores envolvidos: User1.

Passos efetuados para a realização do teste:

1 - Para autorizar o User1 a efetuar consultas na tabela Department, é então criada uma *role* para esse efeito, sendo o primeiro passo descrito na Figura 38.

[33] https://msdn.microsoft.com/en-us/library/ms181362(v=sql.110).aspx

Figura 38 - Opção para criar uma nova *database role*

Na Figura 39 encontra-se a caixa de configuração da nova *role*. A esta vai ser atribuído o nome de Select_Department, terá como dono o dbo, ficando este campo vazio que por defeito é assumido por quem está a criar a *role*, e como membros o User1.

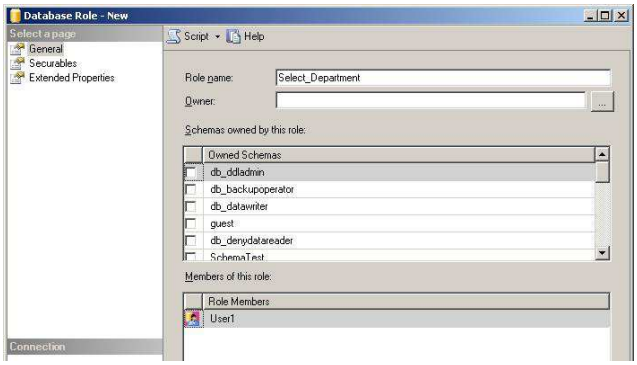

Figura 39 – Menu geral de configuração da *role*

Na Figura 40 apresenta-se a parte de configuração da *role* relativamente aos *securables* que lhe vão pertencer, onde consta a tabela Department do *schema* HumanResources com a permissão de SELECT marcada.

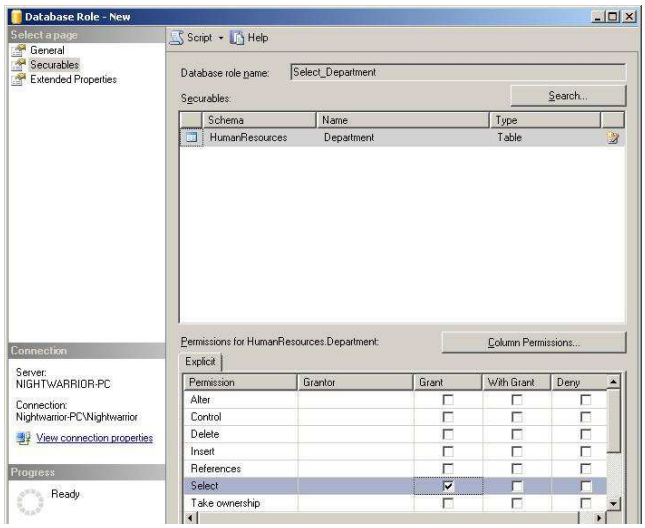

Figura 40 – Seleção dos *securables* na configuração da *role*

2 - Depois de esta *role* estar concluída, o User1 faz novamente uma consulta à tabela Department, e confirma-se que já tem permissão para esta ação.

Existe uma *role* fixa, a *role db_datareader*, que também atribui permissões de leitura, mas sobre todas as tabelas da base de dados, ao contrário da *role* criada e que só atribui permissões de leitura sobre uma única tabela.

Também no menu das propriedades de cada um dos utilizadores, na secção de *Membership*, existe a possibilidade de atribuir *roles* existentes, conforme se mostra na Figura 41.

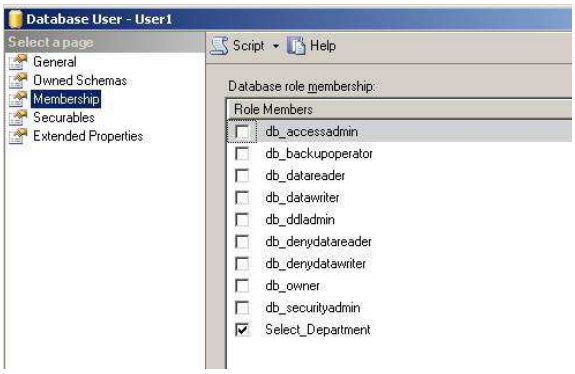

Figura 41 – Seleção de *roles* a atribuir ao User1

Teste 2 (Facilidade na atribuição de permissões): O objetivo deste teste é verificar quais os passos necessários para atribuir, revogar e negar permissões a um utilizador. Uma vez que se está a utilizar *roles*, todas as tarefas de atribuição, revogação e negação de permissões são efetuadas numa *role* já existente, ou numa nova.

Utilizadores envolvidos: User1 e User2.

Passos efetuados para a realização do teste:

1 - Para autorizar o User1 e User2 a consultar e inserir dados nas tabelas HumanResources.EmployeePayHistory e Person.PersonPhone, será criada uma nova *role*, denominada Role2, que vai ter como membros o User1 e User2, como *securables* as tabelas referidas e assinaladas as permissões de SELECT e INSERT, Figura 42 e Figura 43.

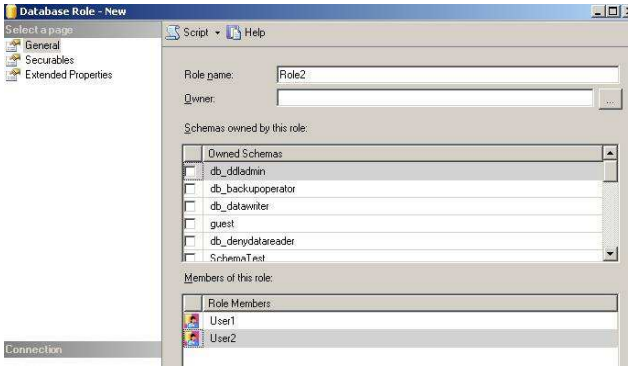

Figura 42 – Atribuição de membros à Role2

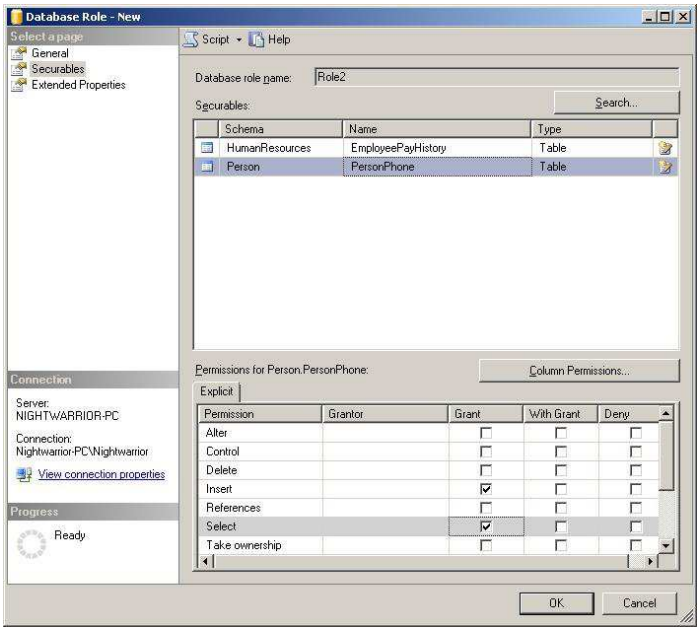

Figura 43 – Atribuição de securables à Role2

2 - Agora vão ser inseridos dados numa das tabelas através do seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
INSERT INTO AdventureWorks2012.Person.PersonPhone VALUES (1705, '123456789', 1,
'2014-01-13');
GO
REVERT;
```

Verifica-se que este utilizador tem permissão para inserir dados na tabela, pois o comando foi executado com sucesso.

3 - O User2 ao efetuar uma consulta sobre a tabela EmployeePayHistory, com o comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User2';
GO
SELECT * FROM AdventureWorks2012.HumanResources.EmployeePayHistory;
GO
REVERT;
```

É obtido o resultado disponível na Figura 44.

	BusinessEntityID	RateChangeDate	Rate	PayFrequency	ModifiedDate
1	1	2003-02-15 00:00:00.000	125,50	2	2008-07-31 00:00:00.000
2	2	2002-03-03 00:00:00.000	63,4615	2	2008-07-31 00:00:00.000
3	3	2001-12-12 00:00:00.000	43,2692	2	2008-07-31 00:00:00.000
4	4	2002-01-05 00:00:00.000	8,62	2	2001-12-22 00:00:00.000
5	4	2004-07-01 00:00:00.000	23,72	2	2004-06-16 00:00:00.000

Figura 44 – Resultado da consulta à tabela EmployeePayHistory

Assim se verifica que para atribuir qualquer permissão ou permissões, sobre um determinado objeto ou objetos, a um utilizador ou a vários, basta apenas uma *role*, desde que as permissões e os objetos sejam idênticos para todos os utilizadores. Para cada novo utilizador criado, basta incluí-lo na *role* e o utilizador ficará com as permissões aos objetos nela definidos. Assim podemos concluir que as *roles* facilitam a atribuição de permissões face ao modelo DAC.

Teste 3 (Verificar o alcance das permissões): O objetivo do teste é permitir a um utilizador a criação de uma tabela na base de dados.

Utilizadores envolvidos: User1.

Passos efetuados para a realização do teste:

1 - O User1 para criar uma tabela na base de dados dentro do *schema* Production, executa o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
CREATE TABLE Production.TableTest (IndexId int, Data nvarchar(50));
GO
REVERT;
```

E verifica que não tem permissões para efetuar esta ação.

```
Msg 2760, Level 16, State 1, Line 1
The specified schema name "Production" either does not exist or you do not have
permission to use it.
```

É então necessário criar uma *role* para conceder permissão ao User1 para criar uma tabela dentro do *schema* Production.

2 - Assim vai ser criada uma nova *role*, a Role3, conforme ilustra a Figura 45.

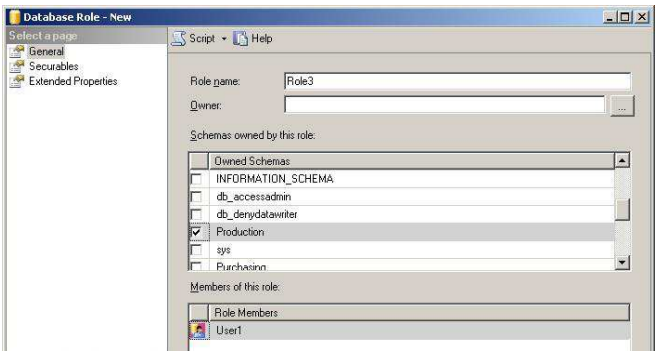

Figura 45 – Configuração da Role3

E esta nova *role* vai ficar como dona do *schema* Production, estando o User1 como membro desta.

3 - Em seguida, o User1 ao executar novamente o comando para criar uma tabela no *schema* Production, verifica que já tem esta ação autorizada.

4 - O User1 vai tentar criar uma tabela como dono da base de dados e verifica que, neste caso, não tem permissão.

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User1';
GO
CREATE TABLE DBTableTest (IndexId int, Data nvarchar(50));
GO
REVERT;
```

```
Msg 2760, Level 16, State 1, Line 1
The specified schema name "dbo" either does not exist or you do not have permission
to use it.
```

5 - Para dar esta permissão ao User1, basta que o administrador coloque este utilizador como membro da *role* fixa db_owner, Figura 46.

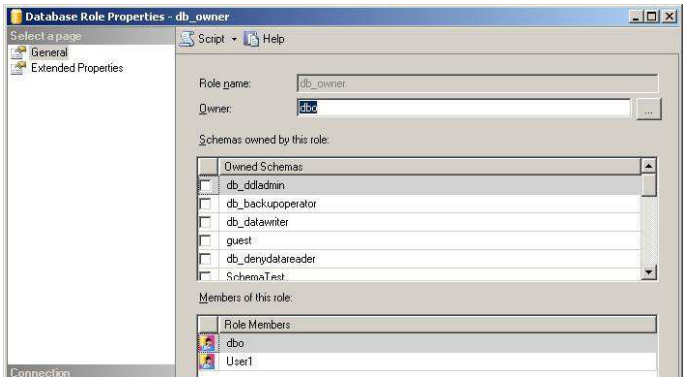

Figura 46 – Atribuição do User1 à *role* fixa db_owner

Desta forma o User1 fica com permissão para criar tabelas na base de dados, sem a utilização de *schemas*.

Verifica-se que as permissões podem ser dadas de forma controlada e diferenciada, existindo a possibilidade de dar permissão a um utilizador para criar uma tabela dentro de um *schema* e não na base de dados. Por último, é possível atribuir permissões através das *roles* de duas formas distintas: usando a *role* fixa db_owner e uma *role* definida para o efeito. A vantagem da *role* fixa é que basta que esta seja atribuída a um utilizador para que este fique com permissões de dono da base de dados, enquanto uma *role* flexível permite atribuir permissões de forma progressiva sobre os objetos que estão contidos na base de dados.

Teste 4 (Atribuição de permissões entre utilizadores): O objetivo é um utilizador atribuir permissões a outros utilizadores sobre os objetos. Para atribuir privilégios sobre objetos aos utilizadores com a opção WITH GRANT, é necessário criar mais uma *role*, que vai conter essa permissão.

Utilizadores envolvidos: User1 e User2.

Passos efetuados para a realização do teste:

1 - Vai ser criada e atribuída a Role4 ao User1, para que este tenha permissões SELECT com a opção WITH GRANT sobre a tabela JobCandidate. De salientar que o User1 deixou de ser membro da *role db_owner*, que lhe tinha sido atribuída no teste anterior. Desta forma o User1 vai poder atribuir permissão para efetuar SELECT a outros utilizadores sobre essa tabela.

2 - Em seguida o User1 vai atribuir permissão ao User2 para efetuar consultas na tabela JobCandidate, com o comando:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
GRANT SELECT ON AdventureWorks2012.HumanResources.JobCandidate TO User2 AS Role4;
GO
REVERT;
```

3 - O User2 vai efetuar o comando de consulta à tabela.

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User2';
GO
SELECT * FROM AdventureWorks2012.HumanResources.JobCandidate;
GO
REVERT;
```

Tendo assim o User2 permissões para consultar a tabela JobCandidate.

Conclui-se que é possível atribuir permissões entre os utilizadors com a opção WITH GRANT através da utilização de *roles*.

Teste 5 (Revogação das permissões): O objetivo é revogar as permissões anteriormente atribuídas por um utilizador aos outros utilizadores.

Utilizadores envolvidos: User1 e User2.

Passos efetuados para a realização do teste:

1 - O User1 vai deixar de ser membro da Role4, de forma a verificar como fica a permissão de SELECT atribuída por este ao User2.

2 - O User2 vai efetuar novamente uma consulta à tabela JobCandidate e verifica que ainda tem autorização, apesar de o User1 já não a ter.

3 - Agora o User1 vai revogar a permissão atribuída ao User2, mas verifica que já não tem essa possibilidade, uma vez que já não pertence a Role4.

```
USE AdventureWorks2012
GO
EXECUTE as USER='User1';
GO
REVOKE SELECT ON AdventureWorks2012.HumanResources.JobCandidate TO User2 AS
Role4;
```

```
GO
REVERT;

Msg 15151, Level 16, State 1, Line 1
Cannot find the user 'Role4', because it does not exist or you do not have
permission.
```

Para que o User1 possa revogar a autorização atribuída ao User2, é necessário que seja novamente membro da Role4, caso contrário esta permissão não pode ser revogada pelo User1. Neste caso, a revogação de permissões é efetuada com *noncascading*, ao contrário do teste 7 do capítulo 4 que só permite revogar as permissões quando todas as que foram atribuídas a partir destas se encontram também revogadas.

Teste 6 (Prevalência das permissões atribuídas por *roles*): O objetivo é verificar qual a permissão que prevalece, se a negação, se a atribuição no uso de *roles*. Para tal vão ser criadas duas *roles*, uma com uma atribuição de permissão sobre um objeto e outra com a negação dessa permissão, sendo estas duas *roles* atribuídas posteriormente ao mesmo utilizador pelo administrador.

Utilizadores envolvidos: User3.

Passos efetuados para a realização do teste:

1 - Vão ser criadas duas *roles*, a Role5 que concede permissão para efetuar consultas na tabela CreditCard, e a Role6 que nega essa permissão.

2 - O User3 vai ser membro da Role5 que concede a permissão, executa o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User3';
GO
SELECT * FROM AdventureWorks2012.Sales.CreditCard;
GO
REVERT;
```

E verifica que tem permissão para consulta.

3 - O User3 ao ser também membro da Role6, que nega essa permissão, DENY, vai efetuar novamente o comando de consulta e verifica que a permissão já está negada.

```
Msg 229, Level 14, State 5, Line 1
The SELECT permission was denied on the object 'CreditCard', database
'AdventureWorks2012', schema 'Sales'.
```

Apesar de estas *roles* terem permissões contrárias, podem ser atribuídas ao mesmo utilizador. Verifica-se assim que prevalece sempre a *role* que tem a negação, em relação à outra *role* que tem a atribuição de permissões.

Teste 7 (Resolução de conflitos de permissões): O objetivo deste teste é verificar como são resolvidos os conflitos de permissões. No entanto não vai ser testado para o caso da atribuição de duas *roles* com permissões inversas porque essa situação já foi analisada no teste anterior, vai sim ser testada a situação em que o administrador atribui a um utilizador uma permissão através de uma *role* sobre um objeto sendo esta negada por um comando DENY e vice-versa.

Utilizadores envolvidos: User4.

Passos efetuados para a realização do teste:

1 – Vai ser criada uma *role*, a Role10 que vai atribuir permissão de consulta de leitura sobre a tabela StateProvince do *schema* Person.

2 – De seguida esta role vai ser atribuída ao User4, de maneira a que possa efetuar consultas de leitura na tabela, conforme foi testado.

3 – Vai ser negada a permissão de consulta de leitura ao User4 sobre a tabela StateProvince por parte do administrador, conforme o comando que se segue:

```
USE AdventureWorks2012
GO
DENY SELECT ON AdventureWorks2012.Person.StateProvince TO User4;
GO
REVERT;
```

Este comando foi executado com sucesso.

4 – O User4 vai novamente efetuar uma consulta de leitura na tabela StateProvince e verifica que já não tem permissão, conforme mostra a Figura 47.

Figura 47 – Consulta negada ao User4 sobre a tabela StateProvince

5 – Por último, vai ser feito o inverso, o administrador vai negar a permissão através da Role10, atribuída ao User4, para depois lhe atribuir a permissão através do seguinte comando:

```
USE AdventureWorks2012
GO
GRANT SELECT ON AdventureWorks2012.Person.StateProvince TO User4;
GO
REVERT;
```

Este comando foi executado com sucesso.

6 – O User4 ao efetuar novamente uma consulta de leitura à tabela StateProvince, verifica que também tem a permissão negada.

Conclui-se assim que a resolução de conflitos de permissões é sempre resolvida dando prevalência à negação sobre a atribuição, tal como foi também comprovado no teste 10 do capítulo 4.

Teste 8 (Atribuição de permissões de administração): O objetivo deste teste é o administrador atribuir permissões de administração sobre a base de dados a um utilizador, e este utilizador atribuir permissões a outro utilizador.

Utilizadores envolvidos: User4 e User5.

Passos efetuados para a realização do teste:

1 – O administrador vai colocar o User4 como membro da *role* fixa *db_securityadmin*, ficando este utilizador com permissões de administração na base de dados.

2 – De seguida o User4 vai atribuir permissão de consulta de leitura sobre a tabela Vendor ao User5, com o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE as USER='User4';
GO
GRANT SELECT ON AdventureWorks2012.Purchasing.Vendor TO User5;
GO
REVERT;
```

Tendo este comando sido executado com sucesso.

3 – Neste passo já o User5 tem permissões para efetuar consultas de leitura na tabela Vendor conforme se mostra na Figura 48.

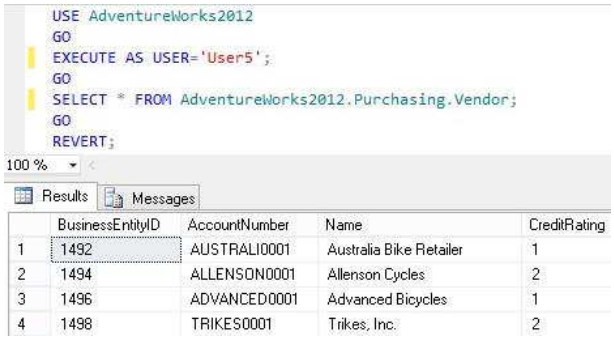

Figura 48 – Consulta de leitura do User5 à tabela Vendor

Assim se pode concluir que o administrador pode delegar permissões de administração a outros utilizadores, para que estes possam por si atribuir permissões a outros utilizadores.

Teste 9 (Revogação de permissões de administração): O objetivo deste teste é o administrador revogar as permissões de administração ao utilizador a que tinha concedido no teste anterior, e verificar como ficam as permissões atribuídas pelo utilizador a outros utilizadores.

Utilizadores envolvidos: User4 e User 5.

Passos efetuados para a realização do teste:

1 – O administrador vai retirar o User4 da *role* fixa *db_securityadmin*.

2 – O utilizador User5, a quem o User4 tinha dado permissão de consulta de leitura sobre a tabela Vendor, vai realizar nova consulta para verificar se ainda tem a permissão que lhe foi atribuída e verifica que ainda continua com a permissão.

Conclui-se deste teste que apesar do utilizador a quem tinham sido concedidas permissões de administração, ficar sem essas permissões, as permissões por este atribuídas a outros utilizadores continuam válidas. Verifica-se assim que a revogação das permissões neste caso é *noncascading*.

Teste 10 (Registo de ocorrências): O objetivo é verificar o registo das ocorrências por parte do SGBD. Este teste é efetuado com recurso a uma ferramenta disponibilizada

pelo MS SQL Server 2012 para auditoria, denominada de *Database Audit Specification*, que já foi analisada no teste 13 do capítulo 4. Desta forma e ao criar uma auditoria para registar os acessos de um utilizador a uma tabela, estes são registados igualmente como no referido teste do capítulo dedicado ao DAC. Neste teste vai ser analisado o registo da atribuição de uma *role* a um utilizador, através do registo de ações do tipo DATABASE_ROLE_MEMBER_CHANGE_GROUP, que regista as atribuições das *roles* aos utilizadores.

Utilizadores envolvidos: User5.

Passos efetuados para a realização do teste:

1 - É criada a Role7 sem permissões atribuídas e somente para fins de atribuição e remoção de utilizadores.

2 - Vai ser criada outra *Audit Action Type* para registar eventos relacionados com a atribuição de utilizadores às *roles*. Esta nova auditoria vai registar ações do tipo DATABASE_ROLE_MEMBER_CHANGE_GROUP, em que são consideradas ações de entrada e saída de utilizadores das *roles*.

3 - Tanto as auditorias do lado do servidor como da base de dados vão ser ativadas.

4 - O User5 vai passar a ser membro da Role7 e em seguida deixar de ser.

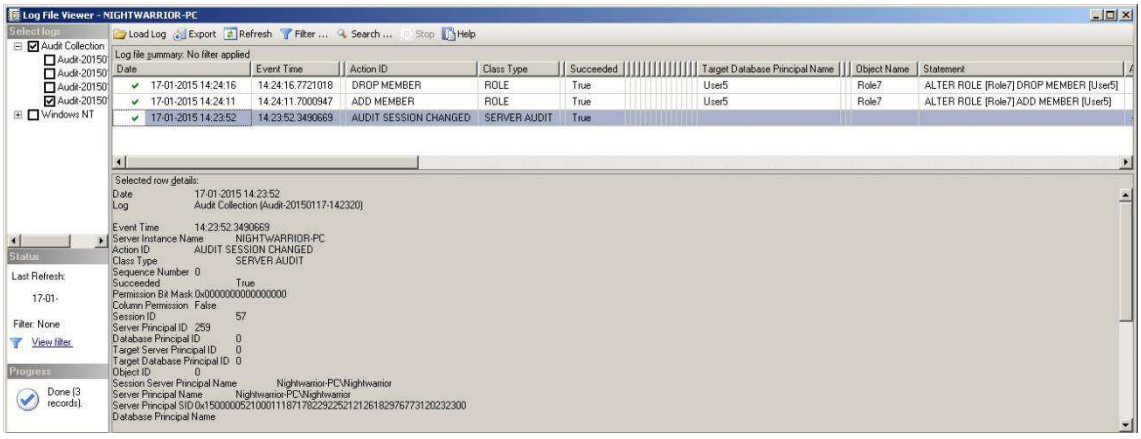

Figura 49 – Visualização das ações efetuadas pelo User5

Como se verifica no registo da auditoria, ver Figura 49, quando o User5 passou a membro da Role7 e deixou de o ser.

Verifica-se através deste teste que o SGBD disponibiliza ao administrador uma ferramenta para registo de atribuição de *roles* aos utilizadores.

Teste 11 (Verificação de permissões atribuídas): O objetivo é verificar que permissões se encontram atribuídas aos diferentes utilizadores e *roles* da base de dados. Neste teste vão ser verificadas as permissões (GRANT e DENY) que estão atribuídas ao User3. Como referido no teste 6 deste capítulo, este utilizador é membro de 2 *roles*, a Role5 e a Role6, em particular iremos verificar para a tabela CreditCard, sendo que para os restantes utilizadores o procedimento será semelhante. Como já foi mencionado anteriormente, o SGBD não dispõe de ferramentas para este fim, mas permite, através de consultas à tabela *fn_my_permissions* a possibilidade de ver as permissões de cada um dos utilizadores.

Utilizadores envolvidos: User3.

Passos efetuados para a realização do teste:

1 - O User3 ao executar o seguinte comando:

```
USE AdventureWorks2012
GO
EXECUTE AS USER = 'User3';
GO
SELECT * FROM fn_my_permissions('AdventureWorks2012.Sales.CreditCard', 'OBJECT');
GO
REVERT;
```

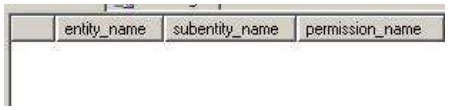

Figura 50 – Permissões do User3 sobre a tabela CreditCard

Verifica que não tem qualquer permissão sobre a tabela CreditCard, Figura 50.

2 - É também necessário verificar se existem outras *roles* a que o utilizador pertença e que possam negar o acesso a um determinado objeto, ao qual foi concedida permissão. O *query* seguinte foi retirado de[34], e permite efetuar uma consulta à tabela *sys.database_role_members* associando as *roles* aos *principals* que são seus membros, da tabela *sys.database.principals*.

```
USE AdventureWorks2012;
GO
```

[34] http://stackoverflow.com/questions/3361778/get-list-of-all-database-users-with-specified-role

```
SELECT rp.name, mp.name FROM sys.database_role_members drm
join sys.database_principals rp ON (drm.role_principal_id = rp.principal_id)
join sys.database_principals mp ON (drm.member_principal_id = mp.principal_id)
```

Este tipo de consulta permite obter as *roles* de um determinado utilizador, ou vice-versa, usando a cláusula WHERE. (WHERE mp.name LIKE 'User3' ou WHERE rp.name LIKE 'Role5'). A Figura 51 mostra todas as *roles* existentes na base de dados e que estão associadas a utilizadores.

	database_role	database_user
1	Select_Department	User1
2	Role2	User1
3	Role2	User2
4	Role3	User1
5	Role4	User1
6	Role5	User3
7	Role6	User3
8	db_owner	dbo

Figura 51 – *Roles* da base de dados atribuídas a utilizadores

3 - Falta ainda saber quais são as permissões, que neste caso são atribuídas através de *roles*, que estão associadas à tabela CreditCard.

Com o comando:

```
USE AdventureWorks2012;
GO
EXEC sp_helprotect @name=CreditCard
```

É obtido o resultado da Figura 52.

	Owner	Object	Grantee	Grantor	ProtectType	Action	Column
1	Sales	CreditCard	Role5	dbo	Grant	Select	(All+New)
2	Sales	CreditCard	Role6	dbo	Deny	Select	(All+New)

Figura 52 – Permissões atribuídas sobre a tabela CreditCard

Assim verifica-se que o User3 pertence à Role5 e à Role6, tendo a Role5 permissão para consultas de leitura na tabela CreditCard e a Role6 a negação desta permissão. Note-se que comparativamente com o teste realizado no DAC com os mesmos objetivos, verificamos que para visualizar as permissões é necessário mais um passo.

Teste 12 (Prevalência de permissões através da execução de *stored procedures*): Este teste tem como finalidade um utilizador efetuar uma ação para a qual não tem permissão, através de uma *stored procedure*. O User4 vai efetuar consultas à tabela EmployeePayHistory, não estando esta ação inicialmente atribuída, porque não foi

concedida essa permissão (como se pode verificar no teste anterior, este utilizador não tem permissões atribuídas), ou no caso seguinte, essa permissão negada.

Utilizadores envolvidos: User4

Passos efetuados para a realização do teste:

1 - Vai ser criada uma *stored procedure* para efetuar consultas à tabela, como se pode ver em seguida.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE sp_selectEmployeePayHistory
AS
BEGIN
	SET NOCOUNT ON;
	SELECT * FROM [HumanResources].[EmployeePayHistory]
END
GO
```

2 - Vai ser criada uma nova *role*, a role8 que vai dar permissão de execução sobre a *stored procedure* criada e que vai ter como membro o User4.

3 - O User4 vai executar a *stored procedure* e efetuar uma consulta à tabela EmployeePayHistory.

```
USE AdventureWorks2012
GO
EXECUTE AS USER='User4'
GO
EXEC sp_selectEmployeePayHistory
GO
REVERT;
```

E assim se verifica que apesar de não ter permissão atribuída para efetuar uma consulta de leitura na tabela, este utilizador consegue realizar esta ação através da *stored procedure*.

4 - É criada mais uma *role*, a Role9, em que é negada a permissão para consultas de leitura na tabela EmployeePayHistory, e o User4 é colocado como membro desta *role*.

5 - O User4 executa novamente a *stored procedure* e a execuação é feita com sucesso, permitindo mais uma vez fazer o SELECT da tabela EmployeePayHistory, mesmo

tendo essa permissão negada. De notar que não consegue fazer o SELECT diretamente à tabela, mas através do *stored procedure* consegue obter o mesmo resultado.

Através das *stored procedures* é possível permitir que os utilizadores efetuem ações sobre os objetos, neste caso à tabela EmplyeePayHistory, sem que lhe tenham sido atribuídas permissões, ou mesmo que estas permissões se encontrem negadas. Os utilizadores podem mesmo não ter conhecimento dos comandos que são executados dentro das *stored procedures* [15, p. 402], no entanto podem executá-los sem quaisquer restrições.

Teste 13 (Verificação de ações permitidas por *stored procedures*): Este teste tem como finalidade verificar as ações que um utilizador pode efetuar através de *stored procedures* e para as quais não tem permissão direta. Como verificado no teste anterior, um utilizador ao ter permissão de execução sobre uma *stored procedure* pode efetuar as ações nesta contidas sem que para isso seja verificado se estas ações se encontram negadas.

Utilizadores envolvidos: User4

Passos efetuados para a realização do teste:

1 - Verificação das permissões atribuídas ao User4 sobre a tabela EmployeePayHistory, conforme Figura 53, e onde se verifica também que este utilizador não dispõe de qualquer permissão sobre a tabela.

Figura 53 – Permissões do User4 na tabela EmployeePayHistory

2 - Também vão ser verificadas se existem *roles* que atribuem permissões sobre a tabela, conforme a Figura 54 e onde se pode apurar que unicamente a Role2 dá permissão para efetuar consultas à tabela, sendo que a Role9 as nega.

```
USE AdventureWorks2012;
GO
EXEC sp_helprotect @name=EmployeePayHistory
```

Results | Messages

	Owner	Object	Grantee	Grantor	ProtectType	Action	Column
1	HumanResources	EmployeePayHistory	Role2	dbo	Grant	Insert	.
2	HumanResources	EmployeePayHistory	Role2	dbo	Grant	Select	(All+New)
3	HumanResources	EmployeePayHistory	Role9	dbo	Deny	Select	(All+New)

Figura 54 – Permissões atribuídas sobre a tabela EmployeePayHistory

3 - Como não existem *roles* que atribuam permissões sobre a tabela, o administrador tem de consultar a que *roles* pertence o User4, conforme a Figura 55.

```
USE AdventureWorks2012;
GO
SELECT rp.name AS database_role, mp.name AS database_user FROM sys.database_role_members drm
join sys.database_principals rp ON (drm.role_principal_id = rp.principal_id)
join sys.database_principals mp ON (drm.member_principal_id = mp.principal_id)
```

Results | Messages

	database_role	database_user
1	Select_Department	User1
2	Role2	User1
3	Role2	User2
4	Role3	User1
5	Role4	User1
6	Role5	User3
7	Role6	User3
8	Role7	User5
9	Role8	User4
10	Role9	User4
11	db_owner	dbo

Figura 55 – *Roles* atribuídas ao User4

O administrador identifica desta forma o User4 como membro da Role8 e da Role9.

4 - O administrador terá, em seguida, de verificar quais as permissões associadas a estas duas *roles*, analisando uma a uma. Como alternativa a este procedimento, o administrador pode analisar que *stored procedures* existem e quais as que efetuam operações que estão negadas a este utilizador e só depois de correr todas as *stored procedures* existentes, verificar que a 'sp_selectEmployeePayHistory' efetua uma consulta à tabela em questão, Figura 56.

```
EXEC sp_helptext 'sp_selectEmployeePayHistory';
```

Results | Messages

	Text
1	CREATE PROCEDURE sp_selectEmployeePayHistory
2	AS
3	BEGIN
4	SET NOCOUNT ON;
5	SELECT * FROM [HumanResources].[EmployeePayHistory]
6	END

Figura 56 – Visualização da sp_selectEmployeePayHistory

5 - De seguida o administrador terá de verificar se o User4 tem permissão de executar a *stored procedure* 'sp_selectEmployeePayHistory', Figura 57.

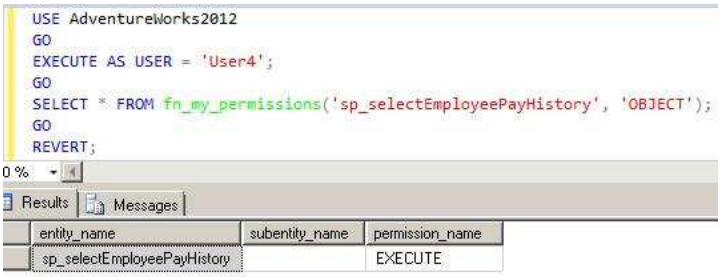

Figura 57 – Permissão do User4 sobre a sp_selectEmployeePayHistory

Só após a conclusão destas tarefas é que o administrador consegue verificar que o User4 pode efetuar uma operação, para a qual tem essa permissão negada, quando usa uma *stored procedure* com a mesma operação.

Esta tarefa parece pouco intuitiva e muito trabalhosa no MS SQL Server 2012, uma vez que não existe um comando para efetuar a ligação entre as operações efetuadas pelas *stored procedures*, os utilizadores que têm permissão de execução nas *stored procedures* e as operações que se encontram negadas. Encontra-se assim dificultada a tarefa de identificar operações que devem estar negadas aos utilizadores, mas que através de *stored procedures* estes podem realizar.

Face aos testes aqui realizados podemos afirmar que o MS SQL Server 2012, no contexto do controlo de acesso não discricionário, permite a atribuição e revogação (gestão) de autorizações de uma forma simples e baseada nas funções / papéis / responsabilidades dos utilizadores. Também foi possível verificar que o SGBD suporta a prevalência da negação, bem como a política *closed world*. Não suporta, no entanto, a revogação recursiva de autorizações, a utilizadores que as obtiveram de um utilizador que viu a sua autorização revogada, ao ter deixado de ser membro de uma *role*. Cabe ainda destacar mais uma vez o papel das *stored procedures*, que pode ser visto como uma forma de contornar permissões revogadas.

6. Métricas de Avaliação

Para além da análise realizada ao MS SQL Server 2012, tendo em vista a verificação dos conceitos dos três modelos de controlo de acesso (DAC, MAC e RBAC), tornou-se necessário analisar o controlo de acesso do referido SGBD seguindo outras métricas. As métricas que vão ser utilizadas são as propostas no Guia para a Avaliação de Métricas nos Sistemas de Controlo de Acesso [7] publicado pelo NIST. Este guia foi escolhido porque é um documento publicado por um organismo credenciado nesta área, sendo um guia para as empresas aplicarem na segurança dos seus processos e aplicações.

As métricas de avaliação estão distribuídas por quatro diferentes propriedades, e de acordo com [7, p. 9] são administração, *enforcement*, desempenho e suporte considerando a avaliação de todo o tipo de sistemas em que seja necessário controlar o acesso. Como tal, nem todas as métricas podem ser aplicadas ao presente trabalho, uma vez que o documento é destinado a todos os sistemas de controlo de acesso, e no presente estudo apenas trabalhamos com um sistema de controlo de acesso de um SGBD ao nível da base de dados.

Assim e como mencionado em [7, p. 1] os sistemas de controlo de acesso têm uma vasta gama de características e capacidades administrativas que podem ter um impacto operacional significativo, tanto do lado da administração como dos utilizadores, bem como na capacidade da organização desempenhar a sua missão. Desta forma, as métricas de avaliação podem verificar as propriedades dos sistemas de controlo de acesso que se dividem por quatro categorias, e tal como temos no documento [7, p. 1]:

1) *Administration is the main consideration of cost;*
2) *Enforcement capabilities are the requirements for AC applications;*
3) *Performance is a major factor for AC usability;*
4) *Support functions allow an AC system to utilize and connect to related technologies so as to enable more efficient integration with network and host service functions.*

Não serão analisadas a totalidade das métricas disponíveis no documento referido, devido ao facto de nos termos concentrado nas métricas ao nível da base de dados. No caso da categoria *performance* avalia as funções que têm impacto com o desempenho do sistema. A categoria *support* engloba funções que podem aumentar a usabilidade e

portabilidade do sistema de controlo de acesso, mas que no entanto, do ponto de vista do NIST não são essenciais [7, p. 9]. Assim, destas quatro categorias vão ser somente estudadas duas delas, administração e *enforment* e dentro destas vão ser estudadas as funções que suportam o controlo de acesso no MS SQL SERVER 2012 através da análise dos seus itens. Para realizar tal análise iremos usar os testes realizados nos capítulos 4 e 5 deste trabalho. A escolha dos itens das métricas baseou-se nas necessidades específicas do sistema de gestão de base de dados que vai ser alvo de estudo e nos modelos de controlo de acesso [3, p. 41].

De acordo com [7] a linguagem XACML é considerada um *standard* para a definição de autorizações em sistemas de controlo de acesso, tendo várias aplicações, como a descrição de políticas, pedidos e respostas a políticas de controlo de acesso [7, p. 5]. Mas o MS SQL Server 2012 não suporta nativamente este tipo de linguagem, sendo necessária a utilização de programas auxiliares para efetuar a conversão desta linguagem para o Transact-SQL [19]. Deste modo não vão ser analisados os itens alusivos ao XACML no MS SQL Server 2012.

Em seguida, vão ser analisadas as funções para avaliar as propriedades que se aplicam ao presente estudo e que se enquadram nas duas categorias selecionadas. Dentro de cada uma das categorias de propriedades há itens de avaliação das métricas. A escala a utilizar para a avaliação do suporte do SGBD às métricas aplicadas vai ser:

Suporta Totalmente - Quando todos os itens avaliados são suportados pelo SGBD;

Suporta Parcialmente - Quando alguns itens avaliados são suportados pelo SGBD;

Não Suporta - Quando nenhum item avaliado é suportado pelo SGBD;

Não Aplicável - Quando os itens não podem ser avaliados, devido a não estarem ao nível da base de dados.

6.1. Administração

Nesta categoria constam as seguintes funções:

6.1.1. *Auditing*

Esta função pode ser analisada através do teste 13 do capítulo 4 e do teste 8 do capítulo 5.

Itens a avaliar:

Registo de falhas no controlo de acesso ao sistema.

Não aplicável ao nível das bases de dados, objeto de estudo, mas sim ao nível do servidor.

Registo de *logs* de acessos negados a objetos.

Suporta totalmente, usando os objetos *Audit* e *Audit Specification*, através dos quais são registadas as ações negadas de utilizadores sobre objetos.

Registo de *logs* de acessos concedidos a objetos.

Suporta totalmente usando os objetos *Audit* e *Audit Specification*, através dos quais são registadas as ações efetuadas de utilizadores sobre objetos.

Parametrização dos dados da auditoria para gerir as capacidades do ficheiro.

Não aplicável, pois não é definido na *Audit Specification* (lado da BD) mas do lado do servidor no *Audit*.

Suporta Totalmente - Pode-se concluir que o SGBD suporta totalmente esta função, dado que os itens analisados são suportados, e os que não podem ser aqui avaliados também são suportados, mas do lado do servidor.

6.1.2. *Privileges / capabilities discovery*

Esta função pode ser analisada através do teste 14 do capítulo 4 e do teste 9 e 11 do capítulo 5.

Itens a avaliar:

O sistema mostra capacidades e privilégios.

Suporta parcialmente, permite efetuar *queries* onde são mostradas as permissões dos utilizadores sobre os objetos, e também quais as permissões que um utilizador tem sobre um objeto, no entanto não disponibiliza uma ferramenta simples e prática para identificar as permissões atribuídas dentro da base de dados. Outro facto relevante, que de acordo com a situação detetada no teste 11 do capítulo 5, existem permissões que não são mostradas de forma clara e eficiente, podendo estas ser facilmente ocultadas, como são o caso das *stored procedures*. Um utilizador, desde que disponha de permissão para executar uma *stored procedure*, pode efetuar a ação nesta contida, sem ser mostrado que este utilizador tem permissão para efetuar tal ação, consequentemente esta permissão está "camuflada".

O sistema mostra variáveis das decisões humanas quanto a capacidades e privilégios.

Não se aplica, dado que se assumiu que estas variáveis das decisões humanas estão relacionadas com o mundo exterior e não podem ser consideradas para efeitos das bases de dados.

O sistema mostra variáveis do ambiente para capacidades e privilégios.

Não se aplica, uma vez que se assumiu que estas variáveis de ambiente estão do lado do servidor / sistema operativo e não têm relação com a base de dados.

O sistema mostra a informação de forma gráfica.

Não disponibiliza o "*What you see is what you get*", logo não suporta este item.

Suporta parcialmente - Mostra as permissões diretas dos utilizadores sobre os objetos, apesar de se considerar que as ações que os utilizadores podem efetuar através das *stored procedures* ficam "camufladas".

6.1.3. *Ease of privileges assignments*

Esta função é avaliada com base na maioria dos testes realizados anteriormente, onde são atribuídas, revogadas e negadas permissões a utilizadores.

Itens a avaliar:

Passos necessários para atribuir, alterar ou remover um privilégio.

Como se pode verificar nos testes 2 e 3 do capítulo 4 e do teste 2 do capítulo 5, basta um passo no primeiro caso e dois passos no segundo caso, um para definir a *role* e o outro para atribuir o utilizador. Esta situação surge de este teste ter sido efetuado com um utilizador apenas. No caso de vários utilizadores e estando estes agrupados basta um passo apenas para o caso das *roles*, que é atribuir o utilizador ao grupo. Na outra situação, é um passo por cada utilizador/permissão.

Passos necessários para atribuir, alterar ou remover uma capacidade a uma *subject* ou grupo de *subjects*.

É o mesmo número de passos do item anterior para um utilizador (*subject*). Para um grupo de utilizadores, também é um passo, mas para um controlo de acesso baseado em RBAC, uma vez que em DAC não há suporte para agrupar utilizadores.

Passos necessários para atribuir grupos de *subjects* e relações entre os grupos.

Este item só pode ser avaliado no RBAC, uma vez que em DAC não é possível implementar grupos de utilizadores (*subjects*). No RBAC basta um passo para que um utilizador seja membro de um grupo e que fique com as permissões que estão atribuídas a esse grupo.

Passos necessários para atribuir grupos de objetos e relações entre os grupos.

Tanto em DAC como em RBAC se agrupam facilmente os objetos, no primeiro em *schemas* e no segundo em *roles*, mas também estes objetos podem ser agrupados em *schemas* para depois estes *schemas* serem incluídos nas *roles*. Assim, é um passo para colocar cada objeto dentro do grupo (*role* ou *schema*), podendo em seguida ser colocado o *schema* dentro da *role* em mais um passo.

Passos necessários para atribuir privilégios por herança.

Para atribuir permissões por herança basta que um utilizador fique membro de uma *role* e automaticamente herda as permissões contidas na *role*, ou seja, todas as permissões da *role*, assim como as permissões das *roles* contidas nesta.

Suporta Totalmente - Os itens aqui avaliados, necessitam de um a dois passos para atribuir, alterar ou remover permissões aos diferentes objetos.

6.1.4. *Syntactic and semantic support for specifying AC rules*

Esta função não foi analisada à luz dos testes realizados.

Itens a avaliar:

O sistema de controlo de acessos permite definir expressões lógicas para especificar regras.

Não aplicável, uma vez que não foi necessário aplicar qualquer regra através de expressões lógicas.

O sistema de controlo de acesso aplica lógica de programação para especificação de regras.

Não aplicável, uma vez que o SQL não consta do apêndice A das linguagens de programação em [7].

Não aplicável – Da análise dos itens anteriores, esta função não pode ser avaliada com base nos testes realizados, uma vez que em nenhum destes testes foi utilizada qualquer função descrita nos respetivos itens. Também o SQL não consta como linguagem de programação no apêndice A das linguagens de programação [7].

6.1.5. *Policy management*

Não aplicável - Esta função não pode ser avaliada com os testes realizados porque não foi possível identificar a gestão da política e também porque o estudo não esteve inserido no âmbito de uma organização.

6.1.6. *Delegation of administrative capabilities*

Esta função pode ser avaliada com base nos testes 11 e 12 do capítulo 4 e nos testes 9 e 10 do capítulo 5.

Itens a avaliar:

O sistema de controlo de acesso permite a delegação da administração da política.

Através das *roles*, é possível ao administrador atribuir permissões de administração a outros utilizadores. Também através da atribuição de permissões de controlo sobre um *schema* por parte do seu dono, é possível delegar permissões de administração a outros utilizadores.

Suporta totalmente - O administrador ao colocar um utilizador como membro da *role db_securityadmin*, está a atribuir a este permissões de administração sobre a base de dados no caso do RBAC. No DAC, através da atribuição de permissões de controlo sobre os *schemas*.

6.1.7. *Flexibilities of configuration into existing systems*

Não aplicável - O estudo esteve somente ao nível da base de dados.

6.1.8. *The horizontal scope (across platforms and applications) of control*

Não aplicável - O estudo esteve somente ao nível da base de dados.

6.1.9. *The vertical scope (between applications, DBMS, and OS) of control*

Não aplicável - O estudo esteve somente ao nível da base de dados.

6.2. *Enforcement*

Nesta categoria constam as seguintes funções:

6.2.1. *Policy combination, composition, and constraint*

Não aplicável, uma vez que o presente estudo não implementou uma política em específico, nem esteve num ambiente em particular, como uma organização.

6.2.2. *Bypass*

Itens a avaliar:

Capacidade de contornar regras da política para decisões críticas no controlo de acesso.

Não aplicável, uma vez que não foi implementada uma política em específico. Mas no entanto as *stored procedures* permitem contornar as regras impostas pela política de controlo de acesso.

É tolerável o risco de contornar as regras da política, no caso do sistema de controlo de acesso o permitir.

Não aplicável, uma vez que não foi implementada uma política em específico.

6.2.3. *Least privilege principle support*

O teste 1 do capítulo 4 será a base para avaliar esta função. Este teste que apesar de ter sido efetuado no capítulo referente ao DAC, assenta numa *role*, denominada de *public database role*, que já foi alvo de análise na secção 3.1.3.

Itens a avaliar:

Possibilidade de implementar este princípio.

Suporta totalmente - Qualquer novo utilizador, por defeito, não tem qualquer privilégio sobre a base de dados, ou tem apenas as permissões que estão atribuídas à *public role*. A ausência de permissão é considerada uma revogação, *closed world policy*.

Possibilidade de implementar este princípio através de restrições.

Não suporta – Não é necessário aplicar restrições, uma vez que este princípio já se encontra implementado por defeito. Por outro lado, o SGBD não suporta as restrições ao nível do controlo de acesso.

Possibilidade de implementar este princípio através de outras especificações.

Não suporta - Não são necessárias outras especificações, uma vez que este princípio já se encontra implementado por defeito. Por outro lado, o SGBD não suporta outras especificações, para além das indicadas, ao nível do controlo de acesso.

Suporta parcialmente - Este princípio está implementado por defeito neste SGBD.

6.2.4. *Separation of Duty (SoD)*

Os testes realizados no capítulo 5 dedicado ao RBAC, servem de apoio à avaliação desta função.

Itens a avaliar:

Capacidade para especificar regras SoD estáticas.

Suporta parcialmente - Permite especificar regras estáticas através de *roles*, de forma a separar as permissões entre os utilizadores. No entanto, não permite implementar exclusividade de permissões na atribuição de *roles*, não sendo possível restringir a atribuição de uma *role* a um utilizador em virtude de outra já se encontrar atribuída.

Capacidade para especificar regras SoD dinâmicas.

Não aplicável, este SGBD não foi testado com políticas que implementem regras SoD dinâmicas, tais como a *Chinese Wall Policy* [1, p. 44]. Também não foi encontrada nenhuma documentação relativa a implementação destas políticas neste SGBD.

Capacidade para especificar regras *historical* SoD.

Não aplicável, este SGBD não foi testado com políticas que implementem regras SoD dinâmicas, tais como o modelo *Clark-Wilson* [1, p. 42].

Suporta parcialmente - Permite separar grupos de permissões através de *roles*, mas não suporta restrições e exclusividade no uso das *roles*.

6.2.5. *Safety (confinements and constraints)*

Esta função vai ser avaliada com recurso ao teste 6, 7, 11 e 12 do capítulo 4 e aos testes 4, 5, 8, 9 e 13 do capítulo 5.

Itens a avaliar:

O sistema tem a capacidade de efetuar verificações de segurança de forma a evitar fugas de permissões.

No teste 6 e 7 do capítulo 4, onde não são utilizadas *roles* e apenas *schemas*, as permissões atribuídas entre utilizadores no teste 6, são todas removidas em cascata entre

eles no teste 7, ou seja, são removidas permissões de um utilizador e removidas as permissões que este tinha atribuído a outros utilizadores, não permitindo a utilização da opção *noncascading*. Por outro lado, nos testes 11 e 12 do mesmo capítulo, as permissões atribuídas com base nas permissões de administração (CONTROL), não são revogadas em cascata como no caso dos testes 6 e 7, aplicando-se para esta situação o *noncascading*.

Como se encontra descrito no teste 4 do capítulo 5, o administrador concedeu a um utilizador, através de uma *role* com opção WITH GRANT, permissão para consulta de leitura sobre uma tabela. Esse utilizador por sua vez concedeu a permissão de consulta de leitura a outro utilizador. No teste 5, o primeiro utilizador que era membro da *role* que lhe permitiu conceder a permissão a um segundo utilizador, deixou de ser membro dessa *role*. Ficou assim o segundo utilizador com a permissão que lhe foi atribuída pelo primeiro utilizador, não podendo esta permissão ser revogada pelo utilizador que a concedeu.

Nos testes 8 e 9 do capítulo 5, verifica-se que tal como nos testes 11 e 12 do capítulo 4, as permissões atribuídas com base nas permissões de administração concedidas, não são revogadas em cascata, aplicando-se neste caso também o *noncascading*.

No teste 13 do capítulo 5 verifica-se que a atribuição de permissões de execução sobre *stored procedures* pode colocar em risco a segurança da informação, uma vez que se verificou que o utilizador tinha a permissão de leitura negada, mas através da execução de uma *stored procedure* pode efetuar essa leitura. Também se verificou através do teste 10 que o SGBD não dispõe de uma ferramenta rápida e eficaz para detetar estas situações.

Suporta parcialmente - Pelo que foi analisado nestas situações pode-se dizer que os mecanismos que o SGBD dispõe não são totalmente eficazes na prevenção de fugas de informação, nomeadamente no uso de *roles* com permissões WITH GRANT e nas permissões atribuídas com base em permissões de administração concedidas.

6.2.6. *Conflict resolution or prevention*

O teste 10 do capítulo 4 e o teste 7 do capítulo 5 servem de base a avaliação desta função.

Itens a avaliar:

Capacidade de prevenir conflitos nas regras da política.

O sistema de controlo de acessos dá sempre prioridade à negação em relação à atribuição.

Capacidade de resolver conflitos nas regras da política.

Os conflitos são sempre resolvidos ao ser dada prioridade à negação.

Capacidade de prevenir conflitos, no caso de existirem várias políticas implementadas.

No caso de existirem várias políticas, é sempre dada prioridade à negação. A única exceção é com os *stored procedures*.

Capacidade de resolver conflitos, no caso de existirem várias políticas implementadas.

Os conflitos são sempre resolvidos ao ser dada prioridade à negação.

Suporta totalmente - O SGBD dá sempre prioridade à negação na prevenção e resolução de conflitos.

6.2.7. *Operational / situational or awarness*

Itens a avaliar:

Capacidade de especificar e aplicar *operational / situational awareness control*.

Não aplicável, este item deve ser avaliado noutros níveis que não o da base de dados.

6.2.8. *Granularity of control*

Itens a avaliar:

Permite configurar a granularidade dos objetos que controla (Baseada nos requisitos da organização e na arquitetura do sistema).

Não aplicável, este item deve ser avaliado noutros níveis que não o da base de dados.

6.2.9. *Expression (policy / model) properties*

Não aplicável, esta função necessita de testes em ambiente organizacional com uma política de controlo de acesso definida. O SGBD em questão também não suporta nativamente linguagens de especificação de regras de controlo de acesso, tais como o XACML.

6.2.10. *Adaptable to the implementation and evolution of AC policies*

Não aplicável, esta função necessita de testes em ambiente organizacional com uma política de controlo de acesso e regras definidas.

6.3. Notas Finais

Como últimas notas, devem ser apontadas algumas dificuldades sentidas na aplicação das métricas de avaliação propostas pelo NIST. Existem itens onde é difícil aplicar uma escala, exemplo disso é a métrica "*Ease of privileges assignments"*, onde os seus itens questionam o número de passos para a realização de algumas tarefas, e não são apontados objetivos ideais, nem aceitáveis para essa quantificação.

Também na métrica "*Privileges / capabilities discovery*" no item referente à avaliação das variáveis de decisões humanas e de ambiente, não existe documentação que possa esclarecer e auxiliar na aplicação da mesma.

Nas métricas propostas não existe nenhuma que avalie a possibilidade de não utilizar a opção *noncascading* aquando da revogação de permissões a utilizadores.

De salientar que não existe documentação disponível com exemplos da aplicação prática destas métricas, onde é aplicada a avaliação dos itens a casos práticos.

De seguida na Tabela 6, são apresentados os resultados das métricas de avaliação por modelo de controlo de acesso implementado, para as funções que puderam ser aplicadas, não se apresentando as restantes.

Funções	DAC	RBAC
Auditing	Suporta totalmente	Suporta totalmente
Privileges / capabilities discovery	Suporta parcialmente	Suporta parcialmente
Ease of privileges assignments	Suporta totalmente	Suporta totalmente
Delegation of administrative capabilities	Suporta totalmente	Suporta totalmente
Least privilegie principle support	Suporta parcialmente[35]	Suporta parcialmente
Separation of Duty	Não suporta	Suporta parcialmente
Safety (confinements and constraints)	Suporta parcialmente	Suporta parcialmente
Conflict resolution or prevention	Suporta totalmente	Suporta totalmente

Tabela 6 – Resultados da aplicação das métricas aos testes

Pretende-se desta forma apresentar os resultados da aplicação das métricas de uma forma condensada, comparando a segurança do SGBD do ponto de vista do NIST, nos dois modelos de controlo de acesso. Observando os resultados é possível concluir que, à luz do documento do NIST, existe uma ligeira vantagem do RBAC sobre o DAC, pois o primeiro suporta a função do *Separation of Duty*.

[35] Através da utilização de *roles*

7. Conclusões e Trabalho Futuro

Este capítulo visa refletir e avaliar o trabalho aqui desenvolvido, bem como sugerir novos caminhos para continuar a avaliar sistemas de controlo de acesso em SGBDs, de acordo com as métricas definidas pelo NIST ou recorrendo a outras possíveis metodologias. Assim, este capítulo encontra-se dividido em três secções, sendo que na primeira apresentamos as contribuições do estudo realizado, na segunda secção damos a conhecer alguns aspetos relevantes que pensamos ser importantes para o aprofundamento do tema. Na terceira secção deixamos algumas sugestões de caminhos a desenvolver para quem desejar continuar com o estudo aqui apresentado e que assim realçamos como possíveis trabalhos futuros.

7.1. Contribuições

Ao olharmos para o trabalho aqui desenvolvido face aos objetivos inicialmente traçados, podemos salientar a implementação de alguns dos principais modelos de controlo de acesso no MS SQL Server 2012 e posterior avaliação destes, através da aplicação das métricas propostas pelo NIST.

Ao longo dos 28 testes realizados, foram implementados dois modelos de controlo de acesso, um discricionário e outro não discricionário e ainda se verificou que um terceiro modelo de acesso não discricionário não pode ser implementado de raíz neste SGBD. Pretendeu-se com esta investigação avaliar o MS SQL Server 2012 no suporte destes três modelos de controlo de acesso, utilizando para isso a base de dados AdventureWorks2012. Assim, e com base nos testes realizados podemos concluir que este SGBD suporta tanto o modelo DAC como o RBAC, não suportando, no entanto, o modelo MAC. Cabe ainda destacar que os conceitos base do controlo de acesso, no contexto das bases de dados, são suportados pelo MS SQL Server 2012, a saber, *Separation of Duties* e o princípio do *Least Privilege*, mas dependerá do modelo escolhido (DAC ou RBAC). Com respeito ao RBAC, e de acordo com a classificação apresentada em [20] o MS SQL Server 2012 suporta *hierarchical* RBAC (*level* 2), ou seja as hierarquias nas *roles*.

Na análise que realizámos para a implementação MAC, verificou-se que este SGBD, tal como já referido não tem suporte de raíz para este tipo de modelo de controlo de acesso, apesar de ser possível implementá-lo através de outras ferramentas que têm de ser instaladas no MS SQL Server 2012, ou então com recurso a *roles*, como já foi referido. Deste facto, podemos concluir que este SGBD não é indicado para a implementação de modelos de controlo de acesso MAC.

Aos modelos de controlo de acesso que implementámos, DAC e RBAC, aplicámos as métricas de avaliação NIST para averiguarmos o nível da segurança prestado por este SGBD. Com efeito, podemos concluir que este SGBD é seguro do ponto de vista das métricas aplicadas, tanto em DAC como em RBAC, mas verificando-se que o modelo RBAC tem uma ligeira vantagem comparativamente ao DAC, como se pode ver na tabela de comparação de resultados da aplicação das métricas. Contudo, queremos salientar que o administrador da base de dados deve ter especial atenção na delegação de permissões de administração, bem como na concessão de permissões a utilizadores para que estes possam eles próprios conceder permissões a outros utilizadores, nomeadamente com a opção *with grant*. Ainda no contexto do NIST, as novas versões do MS SQL Server devem ponderar o suporte ao XACML, tal como é realizado atualmente pelos SGBDs da Oracle.

Deste modo, pensamos ter atingido os principais objetivos propostos para este trabalho, apesar de existir ainda muito para fazer nesta área, tal como referido na secção destinada ao trabalho futuro. O presente estudo podem ser um auxiliar a quem na sua vida profissional ou académica, tem de decidir, desenvolver e implementar a segurança no controlo de acesso aos SGBDs, em particular do MS SQL Server 2012, ou simplesmente queira dar continuidade ao trabalho que aqui apresentamos.

7.2. Outros Aspetos a Considerar

Nos capítulos 4 e 5 foi analisada a aplicação dos modelos discricionários e não discricionários, DAC, MAC e RBAC no MS SQL Server 2012 de uma forma separada. Mas como descrito em [16] desde cedo se pensou que os mecanismos existentes para o RBAC fossem suficientes para implementar os outros modelos clássicos, DAC e MAC.

Segundo [16] a aplicação de um modelo MAC ou LBAC (*Lattice Based Access Control*), é possível através do uso de uma hierarquia de *roles* e a definição de

restrições, do inglês *constraints*. Também a implementação de um modelo DAC está previsto no RBAC, através da atribuição de várias *roles* a cada um dos objetos. Também em [3] é referido que a implementação de um modelo DAC em RBAC é possível, mas bastante difícil, tornando esta tarefa com mais interesse teórico do que prático.

De notar que no presente trabalho, não se optou pela implementação de um modelo DAC com recurso a *roles*, uma vez que se assumiu que a utilização de *roles* era feita em exclusivo para o modelo RBAC.

Do mesmo modo em [21] é apresentada a implementação de um modelo MAC, também através de *roles*, e utilizando um outro software que não o MS SQL Server.

Tendo em consideração todos os inconvenientes que advêm da implementação das políticas DAC e MAC em ambiente RBAC, aliado ao facto de o MS SQL Server não permitir o uso de restrições (do inglês *constraints*) na aplicação de *roles*, não foi aprofundado este tema com maior detalhe, podendo assim ser desenvolvido em futuros trabalhos.

7.3. Trabalho Futuro

Como trabalho futuro, o mais complexo será condensar em algumas linhas o muito que pode ser feito nesta área de vital importância, nos dias de hoje. Assim, e como futuros desenvolvimentos no estudo do controlo de acesso à base de dados, temos a aplicação das métricas de avaliação do NIST a outros SGBDs existentes de forma a alertar os seus administradores de possíveis falhas na segurança do controlo de acesso.

Promover a aplicação do presente estudo num ambiente organizacional, onde possam ser colocadas em prática as propostas aqui deixadas e que tendo em conta o contexto em que foi realizado o estudo, muitas delas podem ficar no abstrato.

Desenvolver a implemetação de modelos DAC e MAC em ambiente RBAC, tal como referido anteriormente, dado não ter sido encontrada nenhuma documentação que refira implementações desse género no MS SQL Server 2012.

Por último, aplicar as outras duas categorias de propriedades de métricas que não foram aqui alvo de análise.

Referências Bibliográficas

[1] D. F. Ferraiolo, D. R. Kuhn e R. Chandramouli, Role-Based Access Control, Norwood: Artech House, Inc., 2003.

[2] E. Bertino e R. Sandhu, "Database Security - Concepts, Approaches and Challenges," *IEEE Transactions on Dependable and Secure Computing, Vol. 2, Nº1,* January-March 2005.

[3] V. C. Hu, D. F. Ferraiolo e D. Kuhn, "Assessment of Access Control Systems," National Institute of Standards and Technology, Gaithersburg, 2006.

[4] R. Elmasri e S. B. Navathe, Fundamentals of Database Systems Sixth Edition, Boston: Addison-Wesley, 2011.

[5] S. D. C. d. Vimercati, S. Foresti e P. Samarati, "Recent Advances in Access Control," em *Handbook of Database Security: Applications and Trends*, Springer, 2008, pp. 1-22.

[6] C. Coronel, S. Morris e P. Rob, Database Systems - Design, Implementation and Management 9th Edition, Boston: Cengage Learning, 2011.

[7] V. C. Hu e K. Scarfone, "Guidelines for Access Control System Evaluation Metrics," National Institute of Standards and Technology, Gaithersburg, 2012.

[8] NIST, "A Report Priviliege (Access) Management Workshop," National Institute of Standards and Technology, Gaithersburg, 2010.

[9] M. Gertz e S. Jajodia, Handbook of Database Security, New York: Springer, 2008.

[10] R. Ramakrishnan e J. Gehrke, Database Management Systems, Third Edition, New York: McGraw-Hill, 2003.

[11] S. Vanamali, "Role Engineering: The Cornerstone of RBAC," ISACA - www.isaca.org, 2008.

[12] U.-e.-. Ghazia, R. Masood e M. A. Shibli, "Comparative Analysis of Access Control Systems on Cloud," em *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Islamabad, 2012.

[13] K. Simmons e S. Carstarphen, Pro SQL Server 2012 Administration, Second Edition, Apress, 2012.

[14] B. Beauchemin, "SQL Server 2012 Security Best Practices - Operational and Administrative Tasks," Microsoft, 2012.

[15] L. Davidson e J. M. Moss, Pro SQL Server 2012 Relational Database Design and Implementation, Apress, 2012.

[16] S. Osborn, R. Sandhu e Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Transactions on Information and System Security, Vol. 3, No. 2,* pp. 85-106, 2000.

[17] S. Demurjian, "Implementation of Mandatory Access Control in Role-based Security System," The University of Connecticut, Storrs, 2001.

[18] W. Rjaibi e P. Bird, "A Multi-Purpose Implementation of Mandatory Access Control in Relational Database Management Systems," IBM Toronto Software Laboratory, Markham.

[19] T. Sorley, "Database Implementation of XACML Role-Based Access Control Specification," California State University, Sacramento, 2013.

[20] R. Sandhu, D. Ferraiolo e R. Kuhn, "The NIST Model for Role-Based Access Control: Towards A Unified Standard," 2000.

[21] S. Demurjian, "Implementation of Mandatory Access Control in Role-based Security System," University of Connecticut, Storrs, 2001.